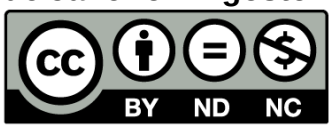
De : Luciano Alves da Silva (lucianopascal@yahoo.com.br)

www.apostilaandroid.net

Rio de Janeiro – Agosto 2012



Creative Commons (CC) - Alguns Direitos Reservados

Aviso sobre esta apostila

Antes de iniciar a leitura deste material, veja esse aviso:

Este material usa a licensa Creative Commons

isto significa

que **ELE PODE SER DISTRIBUÍDO LIVREMENTE**, porém, **SOBRE AS SEGUINTES REGRAS** :

Esse material **NÃO PODERÁ SER COMERCIALIZADO**
Essa material **NÃO PODERÁ SER DEVIRADO**
E todos os créditos do autor **DEVERÃO SER MANTIDOS**

Sobre o Autor da Apostila

Luciano Alves da Silva é Bacharelado em Ciência da Computação pela UNISUAM e Pós-Graduado em Docência do Ensino Superior pelo Instituto A Vez do Mestre (Universidade Cândido Mendes - UCAM). Possui conhecimento e domínio das linguagens de programação Pascal, Java, C/C++, C#, Visual Basic, Delphi, PHP e HTML. Já criou Ambientes de Desenvolvimento Integrado (conhecidos como IDE) como o MakeWare (que trabalha com as linguagens Pascal, C++ e Java) e o AlgoWare (interpretador de algoritmos).

É autor também dos seguintes livros, pela editora AGBOOK

- Aprenda Passo a Passo a Programar em Android – Guia Essencial para Desenvolvedores

- Desenvolvendo Jogos com a Plataforma XNA – Guia para Desenvolvedores.

- Desenvolvendo Jogos com a Ferramenta RPG Maker VX– Guia do Usuário.

Apresentação

O Android é uma plataforma aberta voltada para dispositivos móveis desenvolvida pela Google e atualmente é mantida pela Open Handset Alliance (OHA). Todas as aplicações desenvolvidas para essa plataforma foram criadas com a linguagem Java, o que facilita muitos programadores com conhecimentos em Java (ou de outras linguagens próximas de Java como C++ e C#) a desenvolver aplicações para o Android.

Esta apostila tem por objetivo mostrar de modo fácil como programar na plataforma para dispositivos móveis da Google (Android) usando a IDE Eclipse.

Para quem dedico este material?

Este material é dedicado aos usuários experientes ou iniciantes em programação (tanto para Desktop, Móbile e etc.), que já tenha algum contato com a linguagem Java ou com uma de suas derivadas (como C/C++ ou C#).

Índice analítico

Capitulo 1 Visão geral sobre o Google Android 6

1.1) Introdução 6
1.2) Estrutura Geral da plataforma Google Android.....8
1.2.1) A arquitetura do Android 9
1.2.2) Aplicações 9
1.2.3) Android Runtime.....10
1.2.4) Linux Kernel.....10

Capitulo 2 Instalando e Configurando o Android SDK no Eclipse 11

2.1) Instalando o Eclipse 12
2.2) Instalando o Android SDK e o ADT-Plugin.....14
2.3) Configurando o Android SDK no Eclipse.....23

Capitulo 3 Construindo nossas aplicações no Android 34

5.1) Desenvolvendo uma Calculadora Básica.....34
Aplicação da calculadora em execução.....50
5.2) Desenvolvendo uma aplicação simples de compras.....50
5.3) Desenvolvendo uma aplicação de cálculo de salário.....54
5.5) Desenvolvendo uma aplicação de lista de contatos.....62
5.6) Desenvolvendo uma aplicação que visualiza imagens.....66

Capitulo 4 Trabalhando com mais de uma tela em uma aplicação 76

Capitulo 5 Propriedades e eventos dos componentes trabalhados 84

Widget TextView.....84
Widget EditText.....85
Widget Button.....87
Widget CheckBox/RadioButton 88
Widget ListView.....89
Widget ImageView.....90

Conclusão a respeito do material 91

Capitulo 1 Visão geral sobre o Google Android

1.1) Introdução

Conforme mencionado na apresentação deste material, o Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis, totalmente aberta é livre (Open Source), que foi divulgada em 5 de novembro de 2007. Inicialmente o sistema Android foi desenvolvido pelo Google e atualmente essa plataforma é mantida pela OHA (Open Handset Alliance. Visite o link : http://www.openhandsetalliance.com), um grupo constituído por aproximadamente 84 empresas as quais se uniram para inovar e acelerar o desenvolvimento de aplicações e serviços, com o objetivo e trazer aos consumidores uma experiência mais rica em termos de recursos, menos dispendiosa em ternos financeiros para o mercado móvel.

Um dos primeiros SmartPhones que ofereceu suporte a esse sistema operacional foi o G1 da empresa T-Mobile. Confira na imagem seguinte:

G1 - T-Mobile

Atualmente o sistema Android se encontra hoje disponível tanto em SmartPhones quanto nos famosos Tablets. Confira abaixo alguns dos dispositivos encontramos hoje no mercado com o sistema operacional Android:

SmartPhone Samsung Galaxy

Tablet Motorola XOOM

1.2) Estrutura Geral da plataforma Google Android

O Android SDK é uma ferramenta de desenvolvimento que disponibiliza um conjunto de APIs necessárias para desenvolver aplicações para a plataforma Android, utilizando a linguagem Java.

Vamos conhecer os recursos encontrados nessa plataforma:

- **Application framework:** Permite a reutilização e substituição de componentes ;

- **Dalvik virtual machine:** É uma Máquina Virtual Java (JVM) voltada para dispositivos móveis ;

- **Browser Integrado** baseado no webkit engine ;

- **Gráficos Otimizados** O Android é constituído por bibliotecas 2D e 3D baseada na especificação OpenGL ES 1.0 ;

- **SQLite:** Sistema Gerenciador de Banco de Dados (SGBD) já embutido no Android para guardar dados ;

- **Suporte multimídia:** A plataforma já oferece para áudio, vídeo e formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF) ;

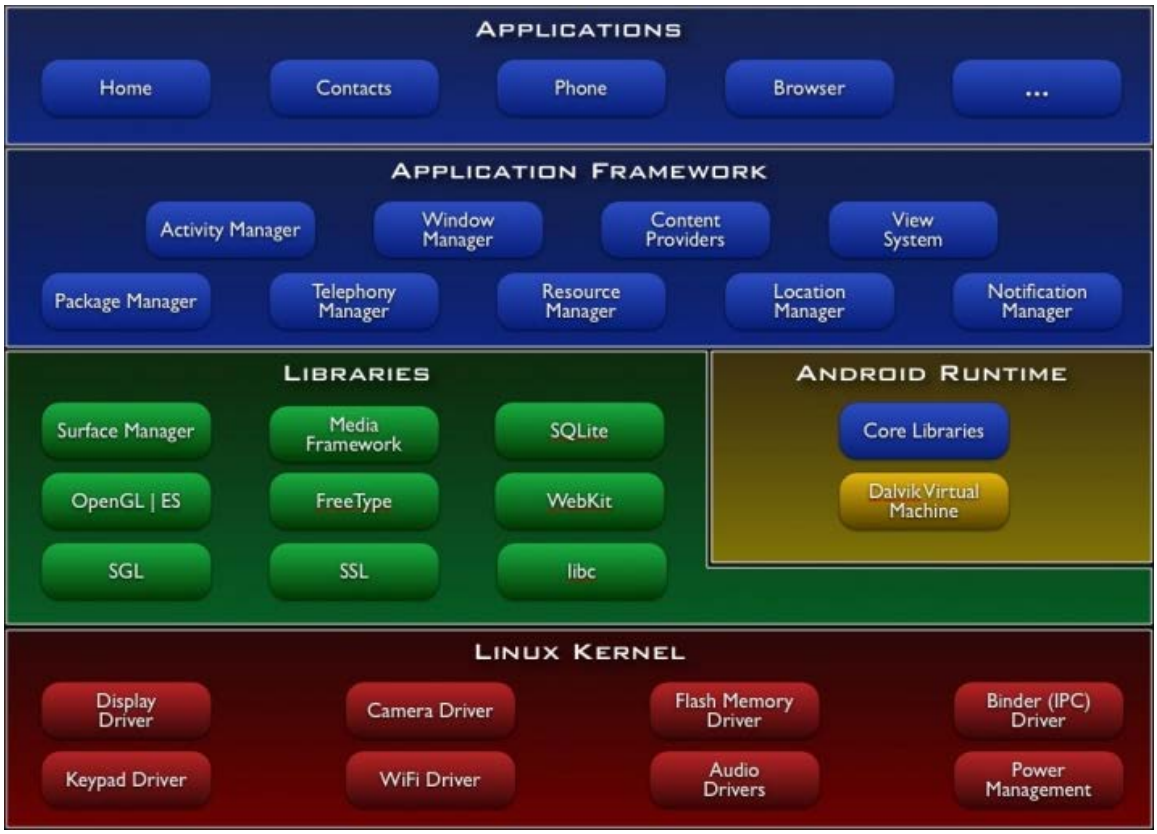
- **Telefonia GSM** (dependente de hardware) ;

- **Bluetooth, EDGE, 3G, e WiFi** (dependente de hardware) ;

- **Câmera, GPS, compasso, e acelerômetro** (dependente de hardware) ;

- **Rico ambiente de desenvolvimento** , incluindo um emulador de dispositivo, ferramentas de depuração, memória, performance e um plugin para o Eclipse (ADT) ;

1.2.1) A arquitetura do Android



Arquitetura geral da plataforma

1.2.2) Aplicações

O Android nos fornece um conjunto de aplicações fundamentais, são elas:

- um cliente de e-mail;
- programa de SMS;
- agenda;
- mapas;
- navegador;
- contatos entre outros.

Todos os aplicativos acima presentes no Android foram desenvolvidos na linguagem de programação Java.

O Android nos fornece um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema. Veja algumas das bibliotecas abaixo:

- **System C library:** Consiste em uma implementação derivada da biblioteca C padrão baseado no sistema (libc) do BSD sintonizada para dispositivos rodando Linux.

- **Media Libraries:** Baseado no PacketVideo's OpenCORE; são as bibliotecas que suportam os mais diversos formatos de áudio e vídeo, incluindo também imagens.

- **Surface Manager:** Responsável pelo acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D;

- **LibWebCore**: Consiste em um web browser engine utilizado tanto no Android Browser quanto para exibições web.

- **SGL – o engine de gráficos 2D**
- **3D libraries**: Uma implementação baseada no OpenGL ES 1.0 APIs; As bibliotecas utilizam aceleração 3D via hardware (quando disponível) ou o software de renderização 3D altamente otimizado incluído no Android.
- **FreeType** – Biblioteca responsável pela renderização de fontes bitmap e vector;

- **SQLite** – Conforme já mencionado, consiste no sistema gerenciador de banco de dados (SGBD) relacional disponível para todas as aplicações.

1.2.3) Android Runtime

O Android é constituído por um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem Java.

Toda aplicação Android roda em seu próprio processo, com sua própria instância da máquina virtual Dalvik. O Dalvik foi escrito de forma a executar várias VMs eficientemente. Ele executa arquivos .dex, que é otimizado para consumo mínimo de memória. A VM é baseada em registros e roda classes compiladas pela linguagem Java que foram transformadas em arquivos .dex, através da ferramenta "dx" incluída no SDK.

O Dalvik VM foi baseado no kernel do Linux para funcionalidades subjacentes como o encadeamento e a gestão de baixo nível de memória.

1.2.4) Linux Kernel

O Android foi projetado em cima da versão 2.6 do kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc. O kernel também atua como uma camada de abstração entre o hardware e o resto do software.

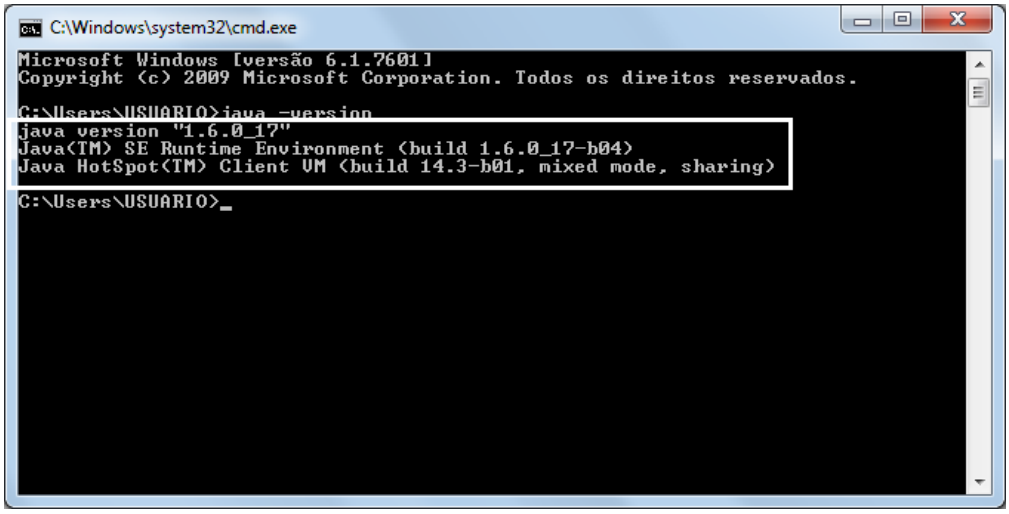
Capitulo 2 Instalando e Configurando o Android SDK no Eclipse

Para a elaboração desse material eu fiz o uso do Eclipse Juno (Eclipse 4.2 para Windows), o SDK do Android Revisão 20 e o plugin do Android para o Eclipse ADT-20.0.0. Qualquer versão (de preferência superior) dos programas citados acima serve. Para que toda essa aplicação funcione é necessário que você tenha instalado, antes de tudo, a Máquina Virtual Java (de preferência a versão 6 ou posterior). Bom, mãos a obra.

Para saber se você possui uma Maquina virtual Java entre no prompt de comando de digite a seguinte linha:

java –version

Se mostrar algo parecido como demonstra a figura seguinte:


C:\Windows\system32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.
C:\Users\USUARIO>java -version
java version "1.6.0_17"
Java(TM) SE Runtime Environment (build 1.6.0_17-b04)
Java HotSpot(TM) Client VM (build 14.3-b01, mixed mode, sharing)
C:\Users\USUARIO>

Maquina Virtual Java instalada no computador

Significa que você possui uma máquina virtual Java instalada no seu computador, caso contrário, instale o JRE. Você pode fazer o download do Java pelo link abaixo:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

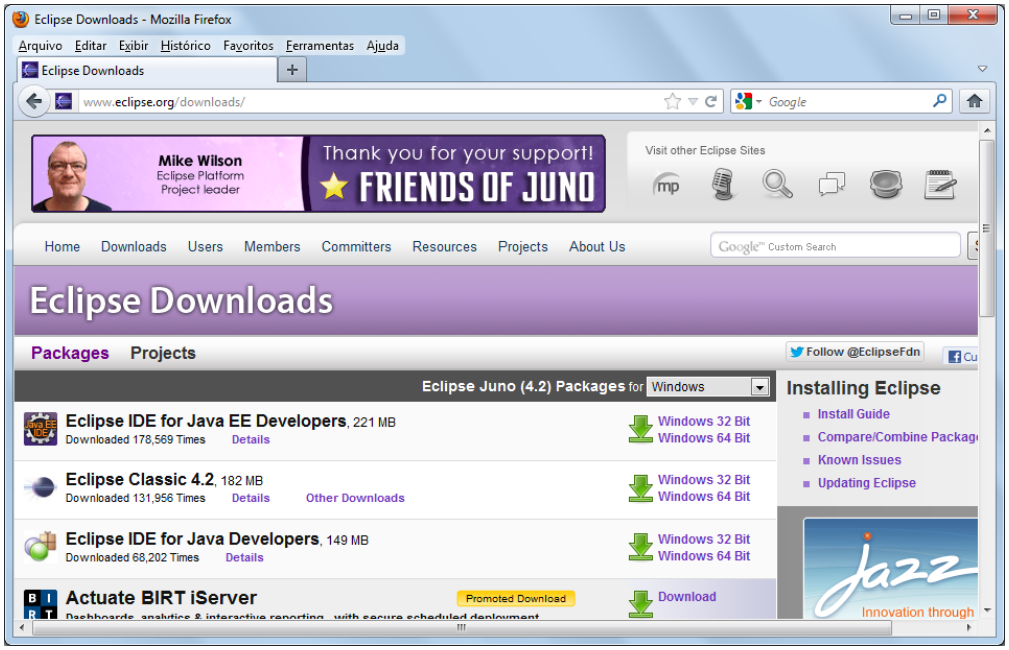
2.1) Instalando o Eclipse

A ferramenta Eclipse Juno (Eclipse 4.2) é a nossa ferramenta de desenvolvimento que utilizaremos para criar nossas aplicações Android. A linguagem que utilizaremos para criar as aplicações será a linguagem Java, na qual a ferramenta Eclipse dá suporte.

Primeiramente, precisamos efetuar o download da ferramenta antes de instalarmos. Para isso, basta entrar no seguinte endereço abaixo:

http://www.eclipse.org/downloads/

Feito isso será carregado a página de download, conforme demonstra a figura abaixo:



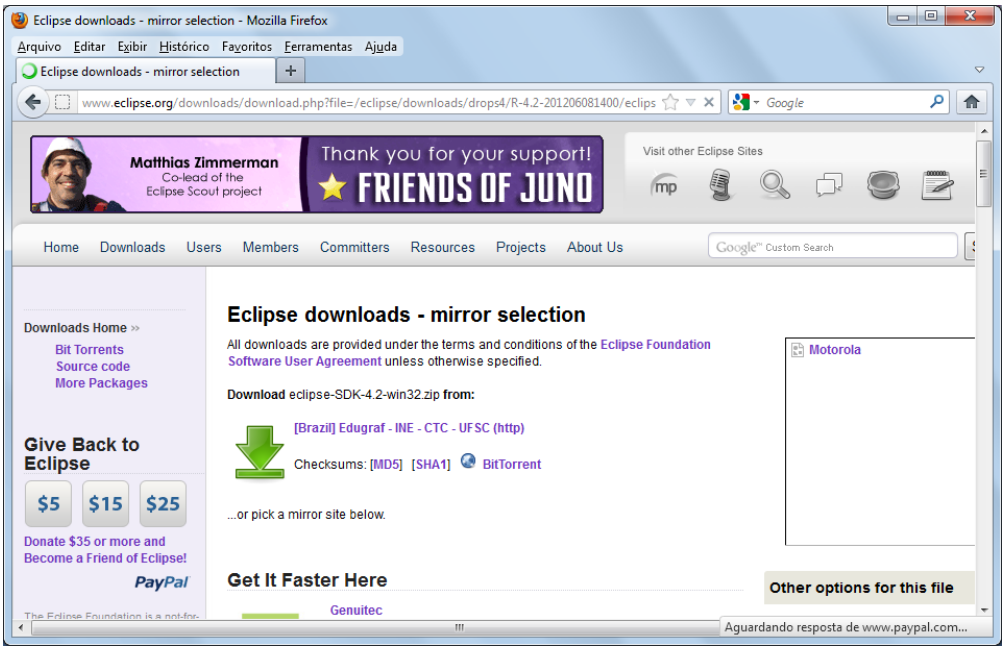
Página de download do Eclipse

Se observamos a página de download, não encontramos nenhum “Eclipse Juno”. Mas então, onde ele se encontra ? O Eclipse “Juno” e conhecido como Eclipse versão 4.2 (ou melhor, Eclipse Classic 4.2). O nome “Juno” simplesmente é um “nome código” (code name) da versão 4.2 (Cada versão do Eclipse tem um nome para cada versão).

O Eclipse Juno (Eclipse Classic 4.2) está disponível para download nas versões 32 bits e 64 bits. Qual das versões devo baixar ? Primeiramente devemos nos certificar da versão do nosso sistema operacional (se ele é de 32 ou 64 bits). Se o sistema operacional for de 32 bits, “obrigatoriamente” terei que baixar a versão 32 bits do Eclipse. Se a versão do seu sistema for de 64 bits, o

mais “viável” é você efetuar o download da versão 64 bits , porém, você pode também baixar a instalar a versão 32 bits do Eclipse em seu sistema de 64 bits sem problemas.

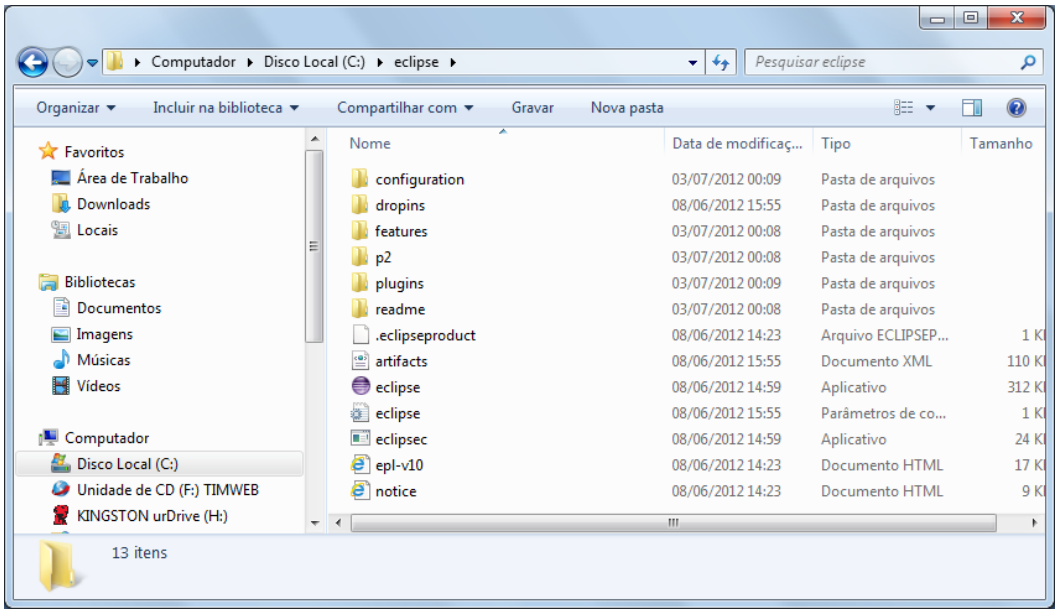
Para a elaboração desta obra, usarei o Eclipse na versão 32 bits. Para isso basta clicar no link com o título “Windows 32 Bit” referente ao Eclipse Classic 4.2. Feito isso vai se abrir a página conforme demonstra a figura abaixo:


Eclipse downloads - mirror selection - Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Eclipse downloads - mirror selection
www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops4/R-4.2-201206081400/eclips
Google
Matthias Zimmerman
Co-lead of the
Eclipse Scout project
Thank you for your support!
FRIENDS OF JUNO
Visit other Eclipse Sites
Home Downloads Users Members Committers Resources Projects About Us
Google™ Custom Search
Downloads Home
Bit Torrents
Source code
More Packages
Give Back to Eclipse
$5 $15 $25
Donate $35 or more and Become a Friend of Eclipse!
PayPal
Eclipse downloads - mirror selection
All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.
Download eclipse-SDK-4.2-win32.zip from:
[Brazil] Edugraf - INE - CTC - UFSC (http)
Checksums: [MD5] [SHA1] BitTorrent
...or pick a mirror site below.
Motorola
Get It Faster Here
Other options for this file
Genuitec
Aguardando resposta de www.paypal.com...

Página de download do Eclipse

Agora basta clicar no link de download para baixar a ferramenta Eclipse.

Depois de efetuado o download da ferramenta Eclipse. Vamos instalar a nossa ferramenta em um local apropriado. Para instalar o Eclipse, basta “descompactá-lo” em um diretório desejado. O diretório de instalação do Eclipse para esse caso será o diretório “raiz” “C:\”. Ao descompactar o arquivo da aplicação, certifique-se de que foi gerado o diretório “C:\eclipse” e que o mesmo apresenta o conteúdo demonstrado na figura abaixo:



Conteúdo do diretório “eclipse”

Se tudo estiver de acordo com a figura acima, sigfinica que os procedimentos foram seguidos, conforme mencionado.

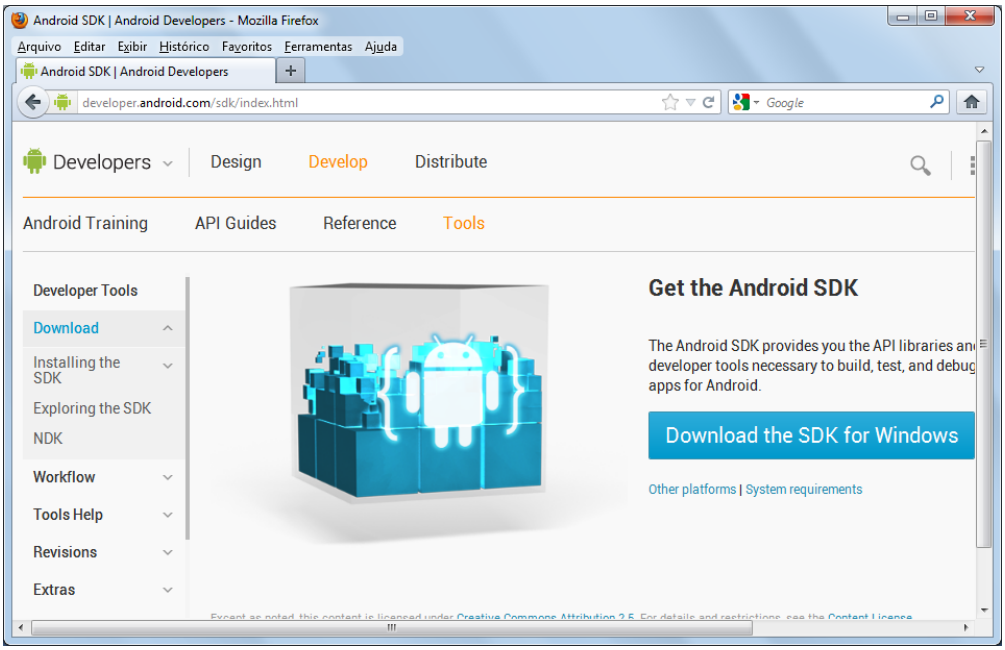
2.2) Instalando o Android SDK e o ADT-Plugin

O Eclipse pelo fato de trabalhar com a linguagem Java , não acompanha nenhum kit de desenvolvimento para Android. Esse kit de desenvolvimento devemos obter no site de desenvolvedores do Android.

Para essa obra estarei utilizando a versão do Android SDK revisão 20. Para efetuar o download devemos visitar o link abaixo:

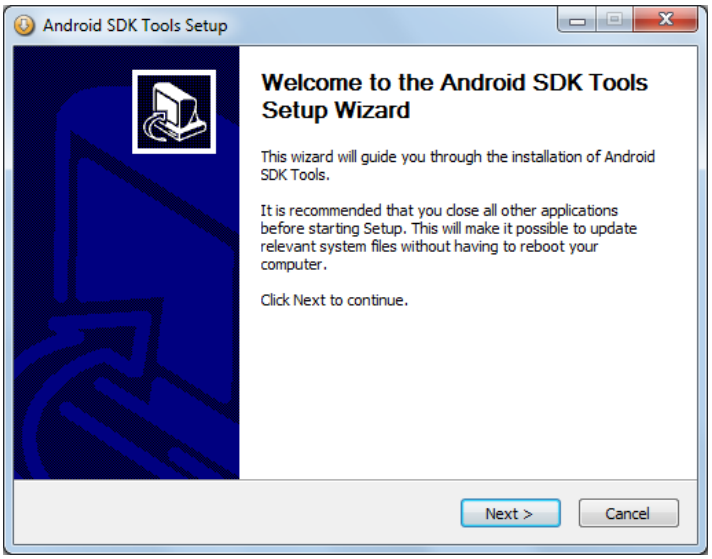
http://developer.android.com/sdk/index.html

Depois de visitar o link será aberta a seguinte página, conforme demonstra a figura seguinte:



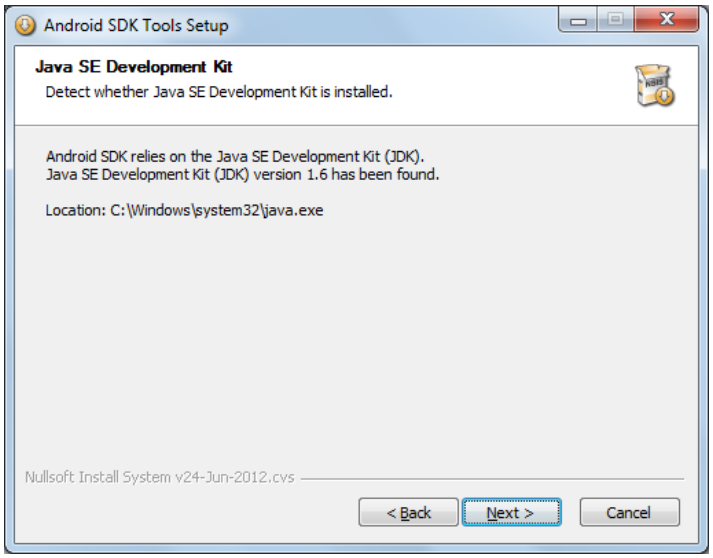
Site do Android – Download do Android SDK

Para efetuar o download do Android SDK, basta clicar no botão “Download the SDK Windows”. Depois de efetuado o download do Android SDK, vamos instalar ele no nosso computador. Execute o utilitário de instalação, conforme você confere na figura abaixo:


Android SDK Tools Setup
Welcome to the Android SDK Tools Setup Wizard
This wizard will guide you through the installation of Android SDK Tools.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

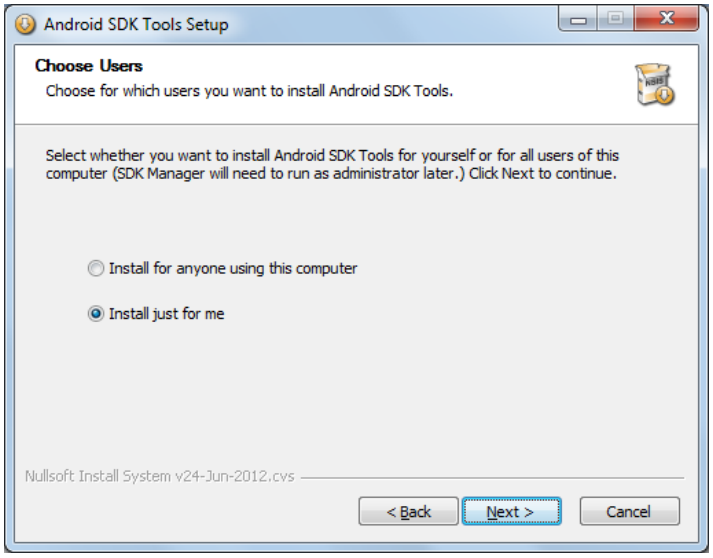
Instalação do Android SDK

Clique em “Next” para continuar. Na próxima seção, será verificado se você tem o Java instalado em sua máquina. Se você tiver, a detecção ocorrerá com sucesso, conforme você pode conferir na figura seguinte:


Android SDK Tools Setup
Java SE Development Kit
Detect whether Java SE Development Kit is installed.
Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.6 has been found.
Location: C:\Windows\system32\java.exe
Nullsoft Install System v24-Jun-2012.cvs
< Back
Next >
Cancel

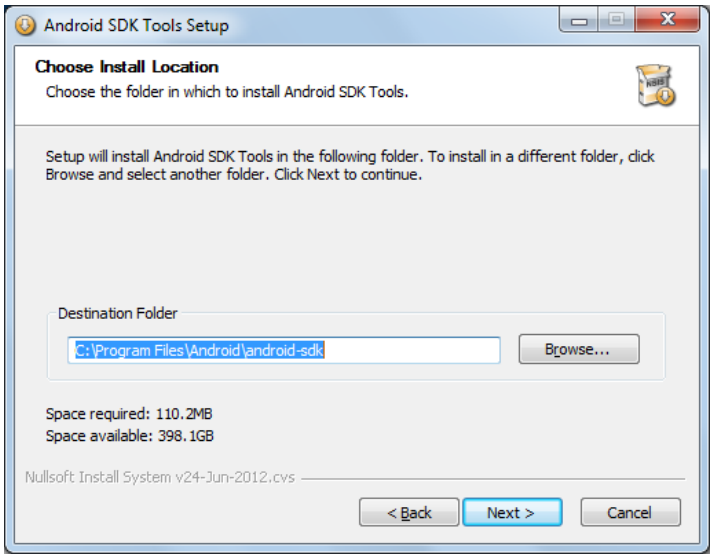
Instalação do Android SDK

Vamos clicar em “Next” para continuarmos com a instalação. Possivelmente ao chegar na próxima tela, será solicitado para quais usuários a instalação será visível, conforme é demonstrado abaixo:


Android SDK Tools Setup
Choose Users
Choose for which users you want to install Android SDK Tools.
Select whether you want to install Android SDK Tools for yourself or for all users of this computer (SDK Manager will need to run as administrator later.) Click Next to continue.
Install for anyone using this computer
Install just for me
Nullsoft Install System v24-Jun-2012.cvs
< Back
Next >
Cancel

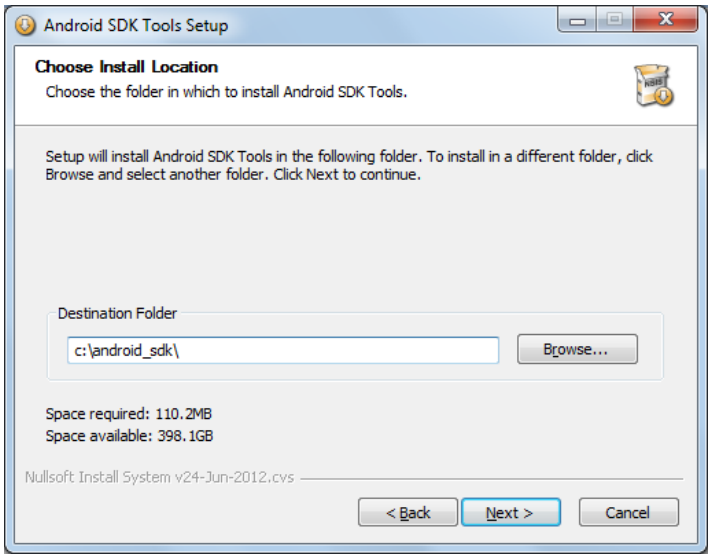
Instalação do Android SDK

Você vai escolher a opção mais apropriada para o seu caso. Depois disso clique em “Next” e na próxima tela vamos definir o diretório de instalação do Android, conforme é mostrado na figura abaixo:


Android SDK Tools Setup
Choose Install Location
Choose the folder in which to install Android SDK Tools.
Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\Android\android-sdk
Browse...
Space required: 110.2MB
Space available: 398.1GB
Nullsoft Install System v24-Jun-2012.cvs
< Back
Next >
Cancel

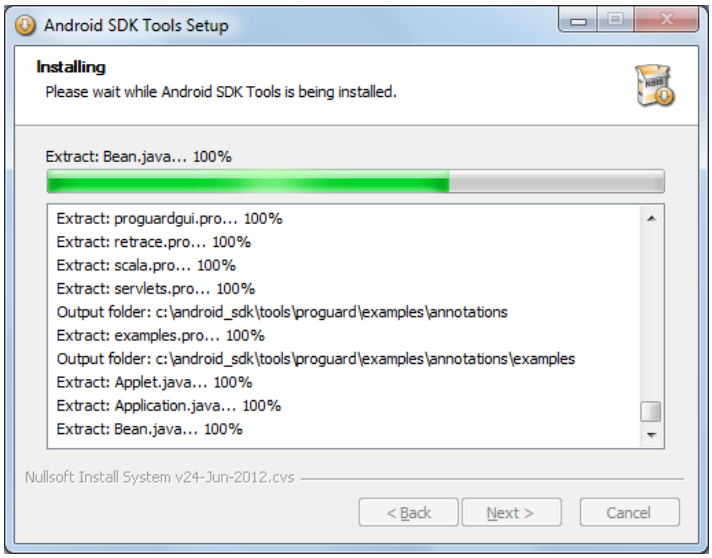
Instalação do Android SDK

Nessa instalação , irei instalar o Android no diretório “c:\android_sdk\”, conforme você confere na figura abaixo:


Android SDK Tools Setup
Choose Install Location
Choose the folder in which to install Android SDK Tools.
Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
c:\android_sdk\
Browse...
Space required: 110.2MB
Space available: 398.1GB
Nullsoft Install System v24-Jun-2012.cvs
< Back
Next >
Cancel

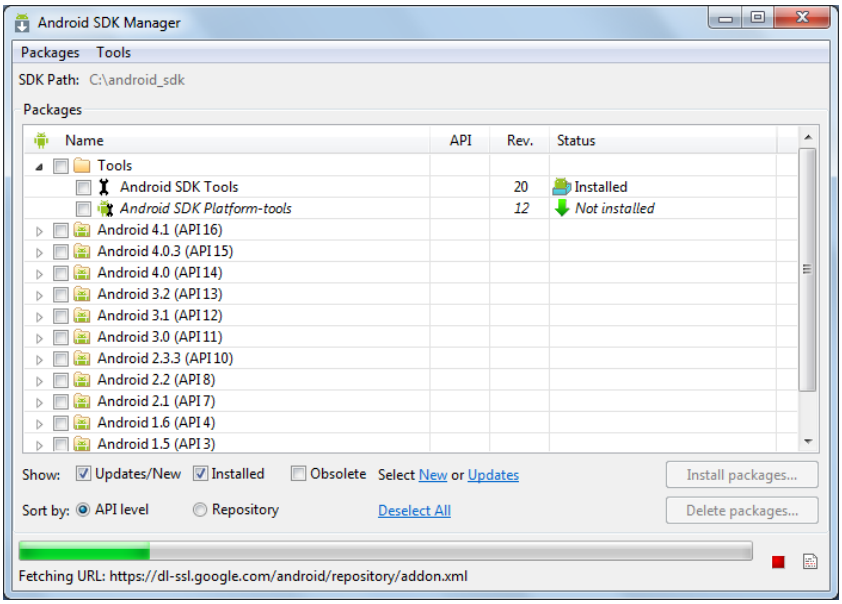
Instalação do Android SDK

Vamos clicar me “Next” para continuar o processo. Na próxima tela simplesmente clique em “Install” para que a instalação seja feita, conforme demonstrado abaixo:



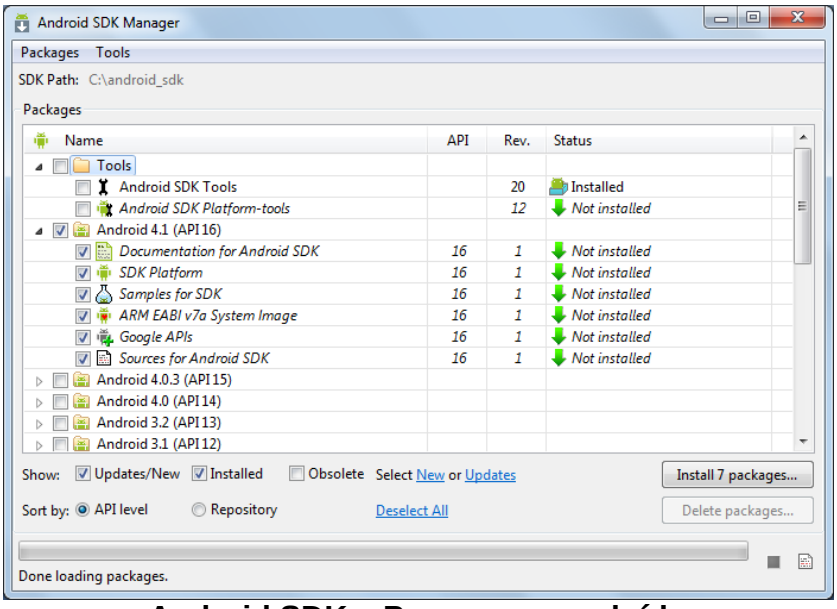
Instalação do Android SDK

Após a instalação ser concluída, execute o Android SDK Manager. Ao carregar a aplicação, o programa irá processar todos os recursos disponíveis, conforme demonstra a figura abaixo:



Android SDK – Buscando atualizações

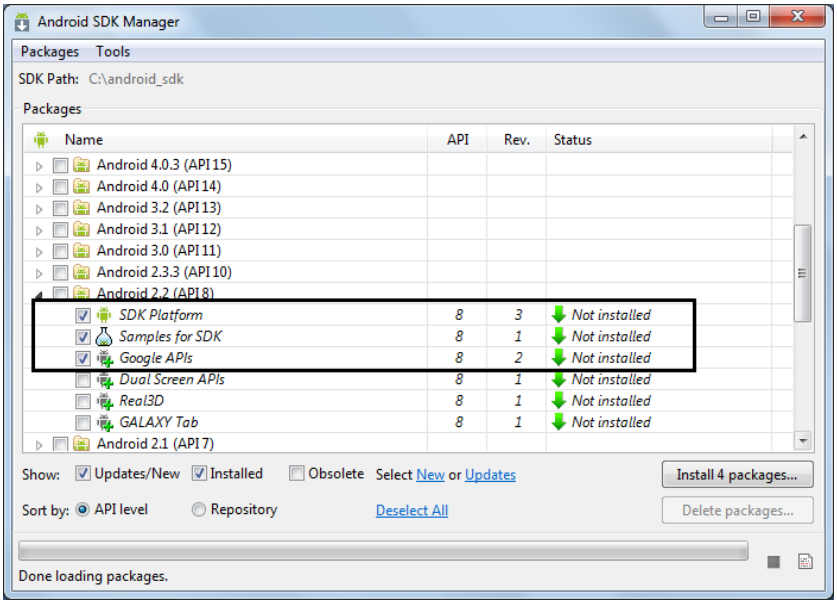
Depois de concluído processo, podemos ver que o Android SDK nos oferece várias plataformas Android e entre outros recursos que podemos utilizar durante o desenvolvimento de nossas aplicações. Por padrão, a opção de download da plataforma Android marcada é a versão 4.1 (Jelly Bean), conforme podemos ver abaixo:


Android SDK Manager
Packages Tools
SDK Path: C:\android_sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 20 Installed
Android SDK Platform-tools 12 Not installed
Android 4.1 (API 16)
Documentation for Android SDK 16 1 Not installed
SDK Platform 16 1 Not installed
Samples for SDK 16 1 Not installed
ARM EABI v7a System Image 16 1 Not installed
Google APIs 16 1 Not installed
Sources for Android SDK 16 1 Not installed
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Show: Updates/New Installed Obsolete Select New or Updates
Install 7 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Android SDK – Processo concluído

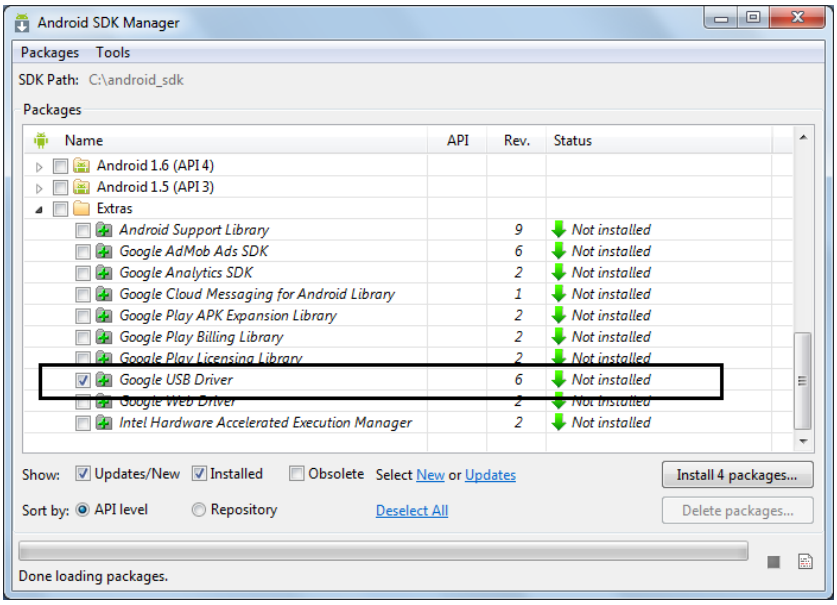
Porém, não iremos trabalhar com essa versão 4.1 da plataforma. Nesta obra iremos trabalhar com a versão 2.2 do Android (Froyo), pelo fato dessa versão do sistema operacional existir na maioria das plataformas móveis existentes (Smartphones e até Tablets). Todas as aplicações desenvolvidas nessa versão do sistema, na prática, são suportadas pelas versões superiores do Android (como o 4.0 e o 4.1).

Para isso vamos desmarcar a opção “Android 4.1 (API 16)” e vamos marcar as seguintes opções do Android 2.2, conforme demonstra a figura seguinte:


Android SDK Manager
Packages Tools
SDK Path: C:\android_sdk
Packages
Name API Rev. Status
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
SDK Platform 8 3 Not installed
Samples for SDK 8 1 Not installed
Google APIs 8 2 Not installed
Dual Screen APIs 8 1 Not installed
Real3D 8 1 Not installed
GALAXY Tab 8 1 Not installed
Android 2.1 (API 7)
Show: Updates/New Installed Obsolete Select New or Updates
Install 4 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Android SDK - Preparando para instalar o Android 2.2

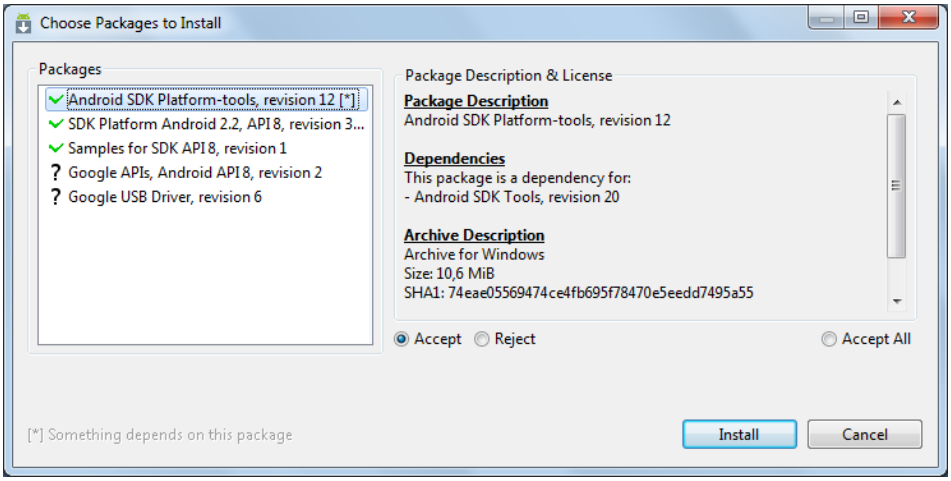
Se rolarmos um pouco abaixo, teremos na seção “Extras” a opção “Google USB Driver” marcado, conforme é demonstrado na figura seguinte:


Android SDK Manager
Packages Tools
SDK Path: C:\android_sdk
Packages
Name API Rev. Status
Android 1.6 (API 4)
Android 1.5 (API 3)
Extras
Android Support Library 9 Not installed
Google AdMob Ads SDK 6 Not installed
Google Analytics SDK 2 Not installed
Google Cloud Messaging for Android Library 1 Not installed
Google Play APK Expansion Library 2 Not installed
Google Play Billing Library 2 Not installed
Google Play Licensing Library 2 Not installed
Google USB Driver 6 Not installed
Google Web Driver 2 Not installed
Intel Hardware Accelerated Execution Manager 2 Not installed
Show: Updates/New Installed Obsolete Select New or Updates
Install 4 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Done loading packages.

Android SDK – Google USB Driver

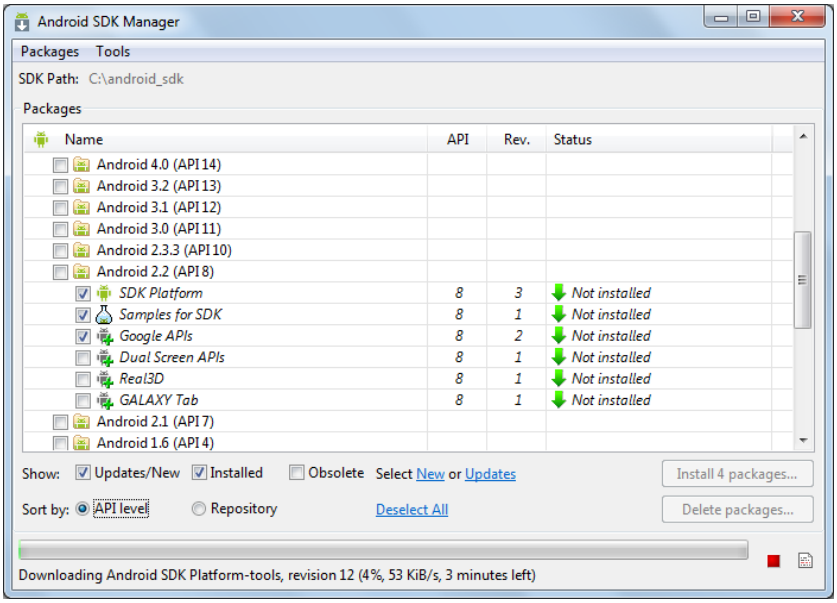
Para que serve esse recurso ? Esse recurso é bastante útil quando queremos testar a nossa aplicação “diretamente” no dispositivo móvel real (sem ser pelo emulador), ou seja, ao invés de rodar a aplicação no emulador, ele irá instalar a aplicação no dispositivo (smartphone) plugado em uma porta USB e irá executá-lo em seguida. Vamos deixar essa opção marcada.

Para instalarmos os recursos selecionados, clique no botão “Install 4 packages” e sem seguida será aberta a seguinte caixa de diálogo:


Choose Packages to Install
Packages
Android SDK Platform-tools, revision 12 [*]
SDK Platform Android 2.2, API 8, revision 3...
Samples for SDK API 8, revision 1
Google APIs, Android API 8, revision 2
Google USB Driver, revision 6
Package Description & License
Package Description
Android SDK Platform-tools, revision 12
Dependencies
This package is a dependency for:
- Android SDK Tools, revision 20
Archive Description
Archive for Windows
Size: 10,6 MiB
SHA1: 74eae05569474ce4fb695f78470e5eedd7495a55
Accept Reject Accept All
[*] Something depends on this package
Install Cancel

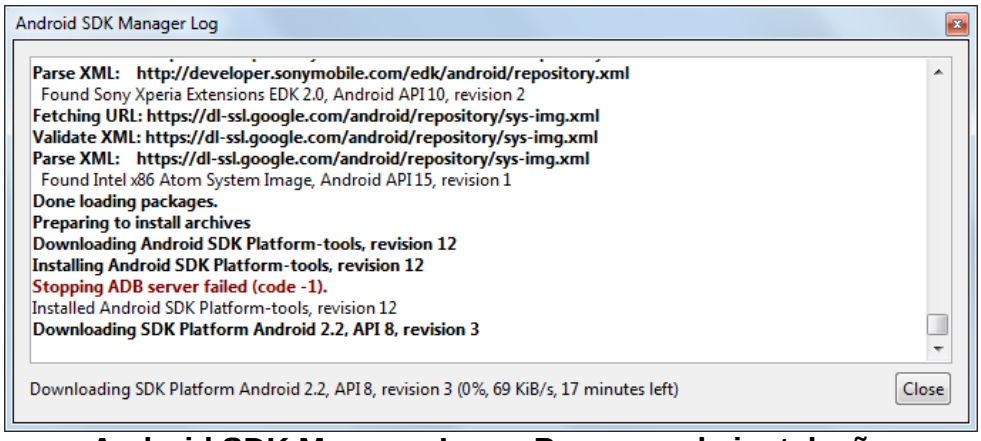
Preparando para instalar

Marque a opção “Accept All” para confirmar a instalação de todos os itens que definimos anteriormente, e em seguida, clique em “Install”. O processo de instalação irá acontecer, conforme você pode ver na figura abaixo:


Android SDK Manager
Packages Tools
SDK Path: C:\android_sdk
Packages
Name API Rev. Status
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
SDK Platform 8 3 Not installed
Samples for SDK 8 1 Not installed
Google APIs 8 2 Not installed
Dual Screen APIs 8 1 Not installed
Real3D 8 1 Not installed
GALAXY Tab 8 1 Not installed
Android 2.1 (API 7)
Android 1.6 (API 4)
Show: Updates/New Installed Obsolete Select New or Updates
Install 4 packages...
Sort by: API level Repository
Deselect All
Delete packages...
Downloading Android SDK Platform-tools, revision 12 (4%, 53 KiB/s, 3 minutes left)

Android SDK – Download do Android 2.2 e de seus recursos

Durante o download do Android e dos seus recursos, deverá surgir a seguinte caixa de diálogo, conforme demonstra a figura abaixo:


Android SDK Manager Log
Parse XML: http://developer.sonymobile.com/edk/android/repository.xml
Found Sony Xperia Extensions EDK 2.0, Android API 10, revision 2
Fetching URL: https://dl-ssl.google.com/android/repository/sys-img.xml
Validate XML: https://dl-ssl.google.com/android/repository/sys-img.xml
Parse XML: https://dl-ssl.google.com/android/repository/sys-img.xml
Found Intel x86 Atom System Image, Android API 15, revision 1
Done loading packages.
Preparing to install archives
Downloading Android SDK Platform-tools, revision 12
Installing Android SDK Platform-tools, revision 12
Stopping ADB server failed (code -1).
Installed Android SDK Platform-tools, revision 12
Downloading SDK Platform Android 2.2, API 8, revision 3
Downloading SDK Platform Android 2.2, API 8, revision 3 (0%, 69 KiB/s, 17 minutes left)
Close

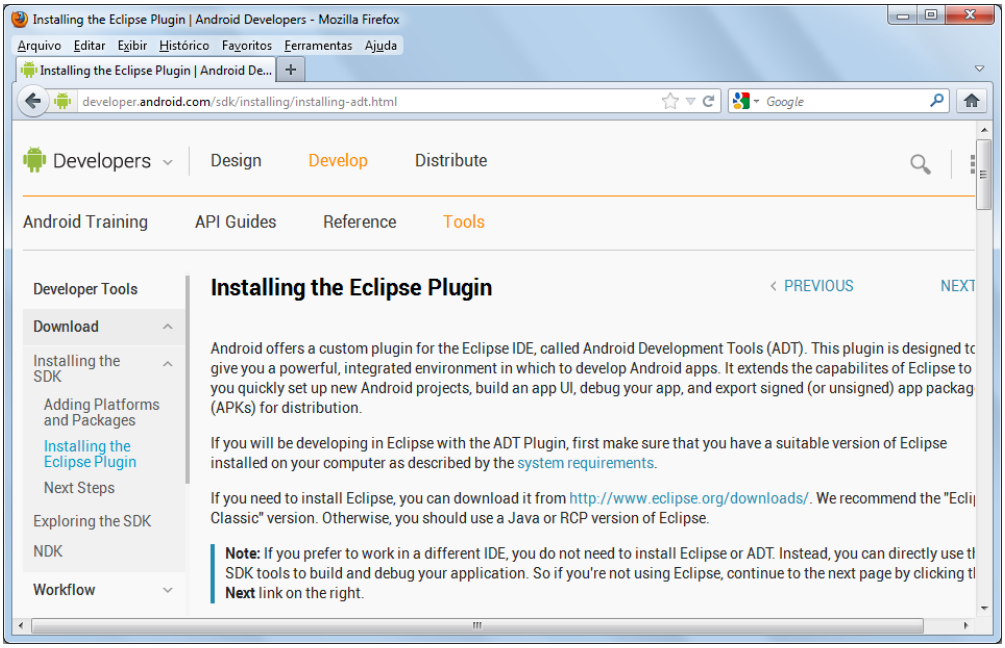
Android SDK Manager Log – Processo de instalação

Aguarde até que o processo de download esteja concluído. Quando o download terminar, feche a caixa de log e em seguida, feche o Android SDK.

Vamos efetuar agora o download do plugin ADT (Android Development Tool), responsável para que possamos programar no Android através do Eclipse. Para efetuar o download, entre no link abaixo:

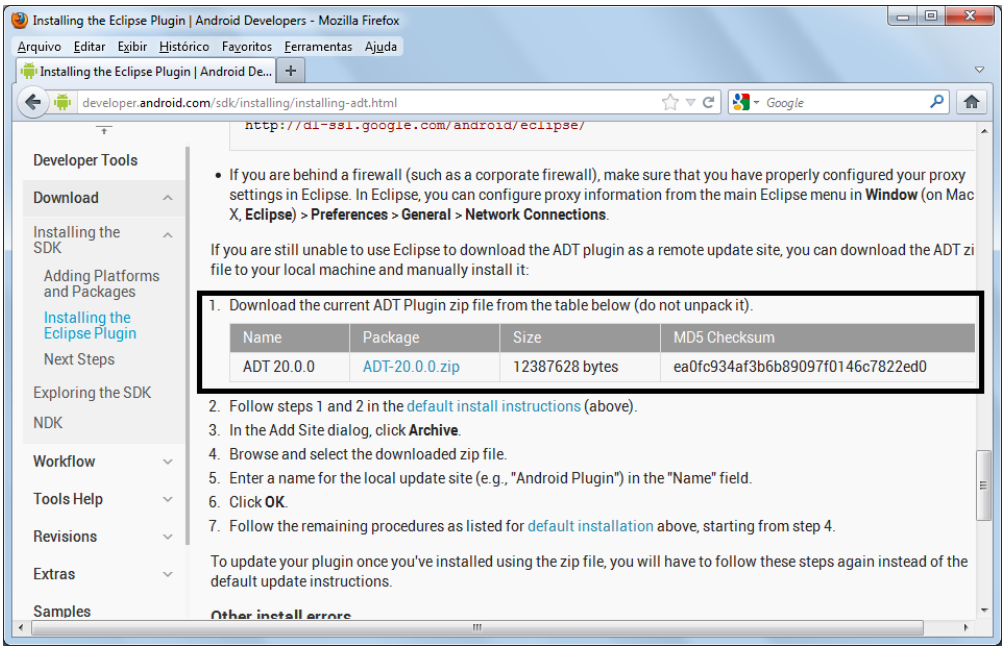
http://developer.android.com/sdk/installing/installing-adt.html

Ao entrar no link acima, será carregado a seguinte página conforme demonstra a próxima figura:


Installing the Eclipse Plugin | Android Developers - Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
developer.android.com/sdk/installing/installing-adt.html
Developers
Design
Develop
Distribute
Android Training
API Guides
Reference
Tools
Developer Tools
Download
Installing the SDK
Adding Platforms and Packages
Installing the Eclipse Plugin
Next Steps
Exploring the SDK
NDK
Workflow
Installing the Eclipse Plugin
PREVIOUS
NEXT
Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin is designed to give you a powerful, integrated environment in which to develop Android apps. It extends the capabilites of Eclipse to you quickly set up new Android projects, build an app UI, debug your app, and export signed (or unsigned) app packag (APKs) for distribution.
If you will be developing in Eclipse with the ADT Plugin, first make sure that you have a suitable version of Eclipse installed on your computer as described by the system requirements.
If you need to install Eclipse, you can download it from http://www.eclipse.org/downloads/. We recommend the "Eclip Classic" version. Otherwise, you should use a Java or RCP version of Eclipse.
Note: If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use th SDK tools to build and debug your application. So if you're not using Eclipse, continue to the next page by clicking th Next link on the right.

Site do Android – Página de download do plugin ADT

Se rolarmos a página abaixo , encontraremos o link para o download do plugin, conforme você confere em seguida:


Installing the Eclipse Plugin | Android Developers - Mozilla Firefox
developer.android.com/sdk/installing/installing-adt.html
http://dl-ssl.google.com/android/eclipse/
Developer Tools
Download
Installing the SDK
Adding Platforms and Packages
Installing the Eclipse Plugin
Next Steps
Exploring the SDK
NDK
Workflow
Tools Help
Revisions
Extras
Samples
If you are behind a firewall (such as a corporate firewall), make sure that you have properly configured your proxy settings in Eclipse. In Eclipse, you can configure proxy information from the main Eclipse menu in Window (on Mac X, Eclipse) > Preferences > General > Network Connections.
If you are still unable to use Eclipse to download the ADT plugin as a remote update site, you can download the ADT zi file to your local machine and manually install it:
1. Download the current ADT Plugin zip file from the table below (do not unpack it).
Name
Package
Size
MD5 Checksum
ADT 20.0.0
ADT-20.0.0.zip
12387628 bytes
ea0fc934af3b6b89097f0146c7822ed0
2. Follow steps 1 and 2 in the default install instructions (above).
3. In the Add Site dialog, click Archive.
4. Browse and select the downloaded zip file.
5. Enter a name for the local update site (e.g., "Android Plugin") in the "Name" field.
6. Click OK.
7. Follow the remaining procedures as listed for default installation above, starting from step 4.
To update your plugin once you've installed using the zip file, you will have to follow these steps again instead of the default update instructions.

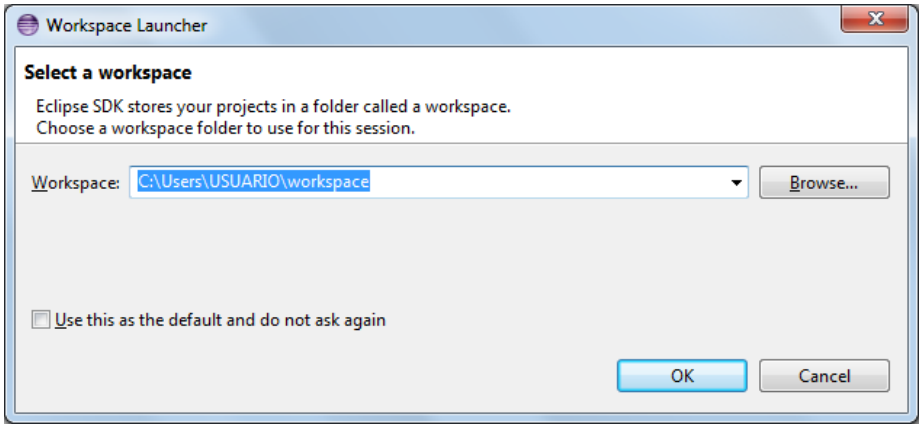
Site do Android – Link do download do plugin

Para efetuar o download basta clicar no local indicado na figura acima.

Depois do download do plugin, coloque-o dentro do diretório “C:\eclipse\”, local onde se encontra instalado a ferramenta de desenvolvimento para Java (Android).

2.3) Configurando o Android SDK no Eclipse

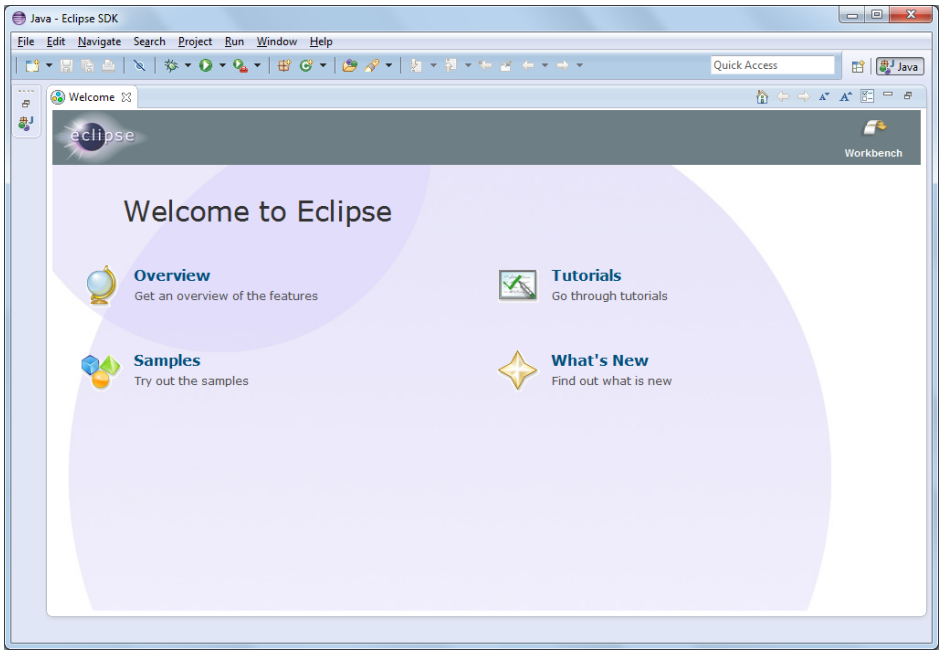
Uma etapa concluída. Agora vamos configurar o Android SDK no Eclipse, de forma que possamos desenvolver nossas aplicações para essa plataforma através do plugin ADT que acabamos de baixar.
Quando você executar o Eclipse pela primeira vez, será solicitado um diretório de “Workspace” (diretório de trabalho), que é o local no qual o Eclipse vai gerar os projetos, conforme você confere na figura seguinte:


Workspace Launcher
Select a workspace
Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace:
C:\Users\USUARIO\workspace
Browse...
Use this as the default and do not ask again
OK
Cancel

Diretório do Workspace

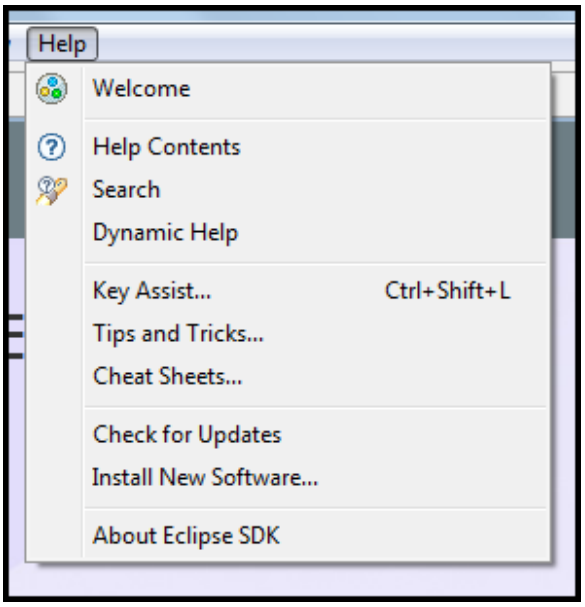
Escolha o diretório desejado para seu “Workspace” e , caso você deseje que ela seja definitivo, marque a opção “Use this as the default and do not ask again”. Depois disso clique em “OK”.

Feito isso, o ambiente de programação Eclipse será carregado, conforme demonstra a figura seguinte:



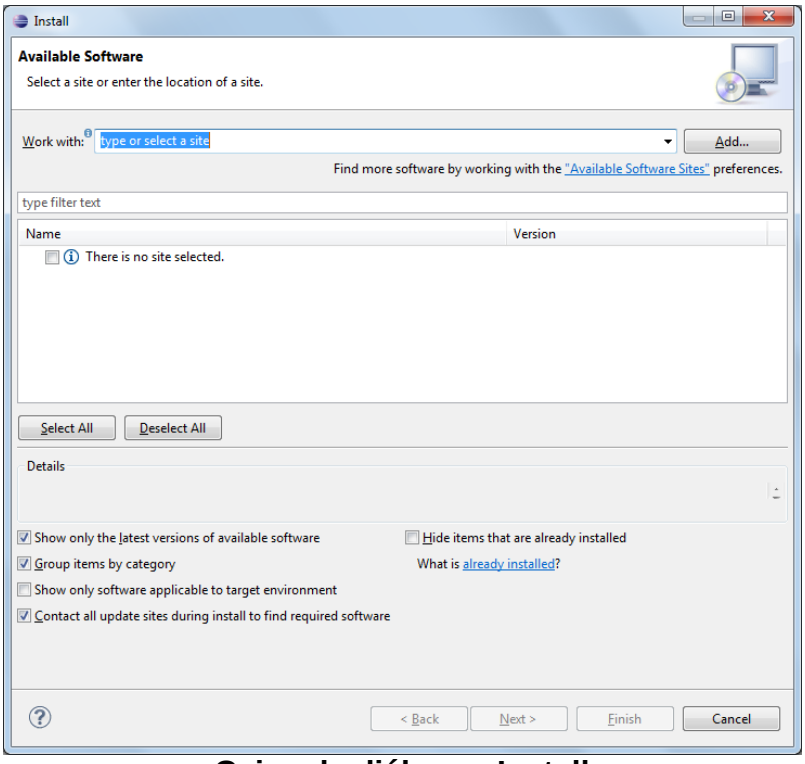
Ambiente de programação Eclipse

Agora vamos instalar o plugin do Android para podemos fazer a interface do Eclipse com o Android SDK, permitindo que a gente desenvolva aplicações para essa plataforma. Vamos no menu “Help” e sem seguida selecione “Install New Software”, conforme demonstra a figura abaixo:



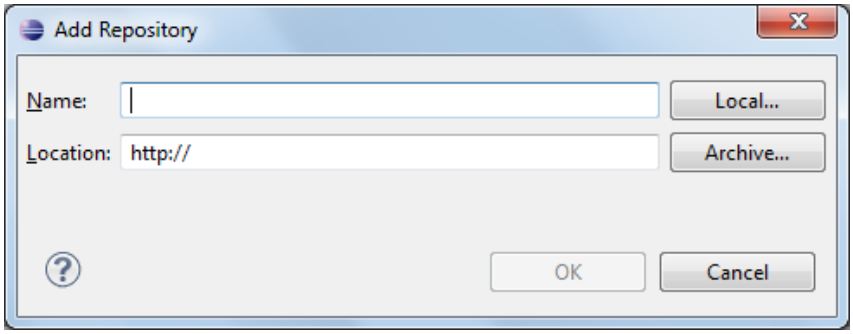
Install New Software

Feito isso será aberta a caixa de diálogo conforme demonstra figura seguinte:


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

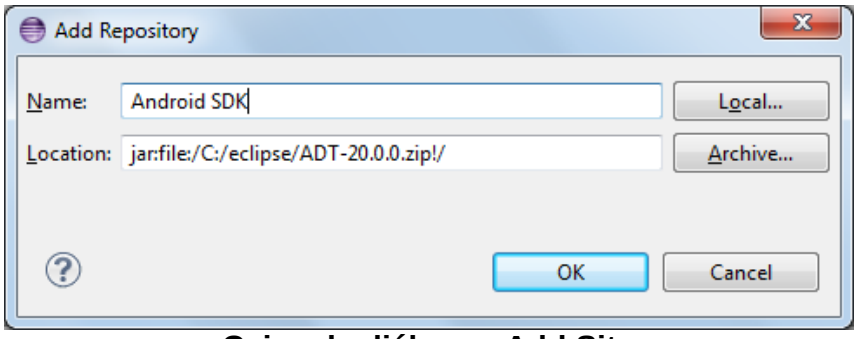
Caixa de diálogo - Install

Para instalarmos o plugin do Android, clique no botão “Add”, e será exibida uma caixa de diálogo conforme mostra a figura abaixo:



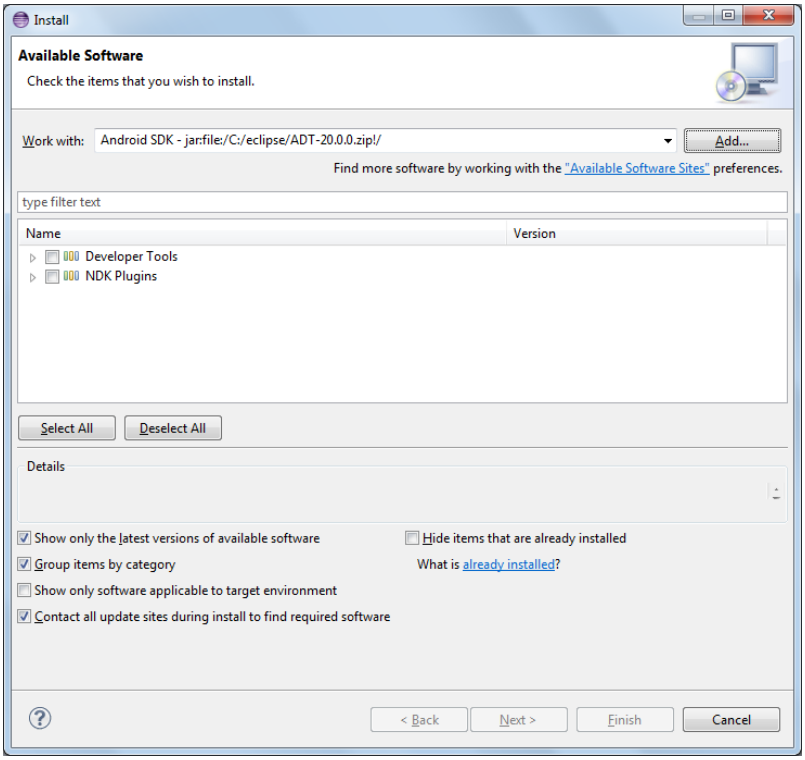
Caixa de diálogo - Add Repository

Agora vamos clicar no botão “Archive” e iremos procurar e selecionar o plugin do Android “A.D.T-16.0.1.zip”. Preencha o campo “Name” como mostra a figura abaixo:



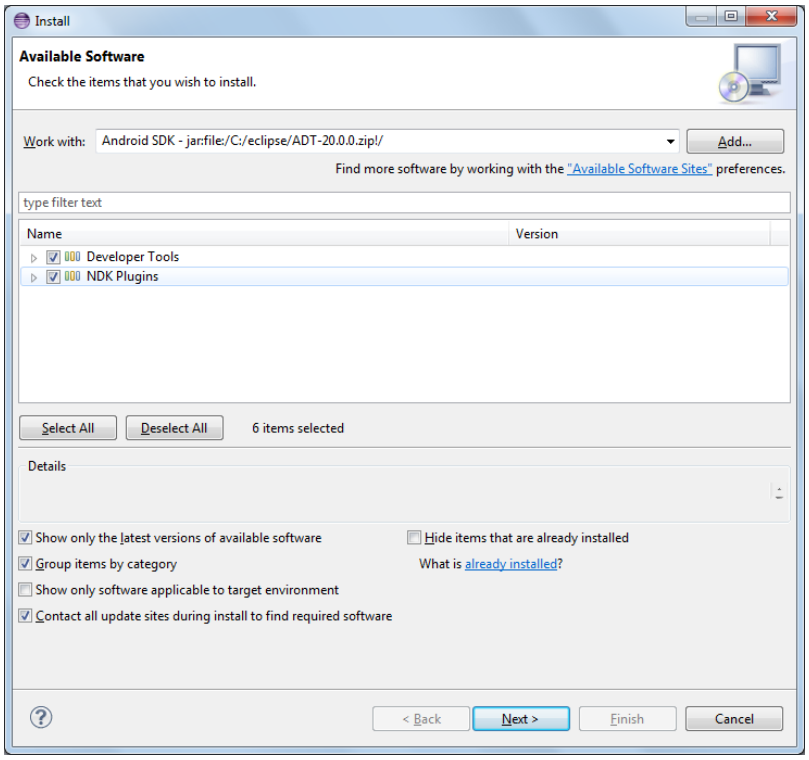
Caixa de diálogo - Add Site

Ao clicar em “OK” será mostrada uma tela, conforme demonstra a figura seguinte:



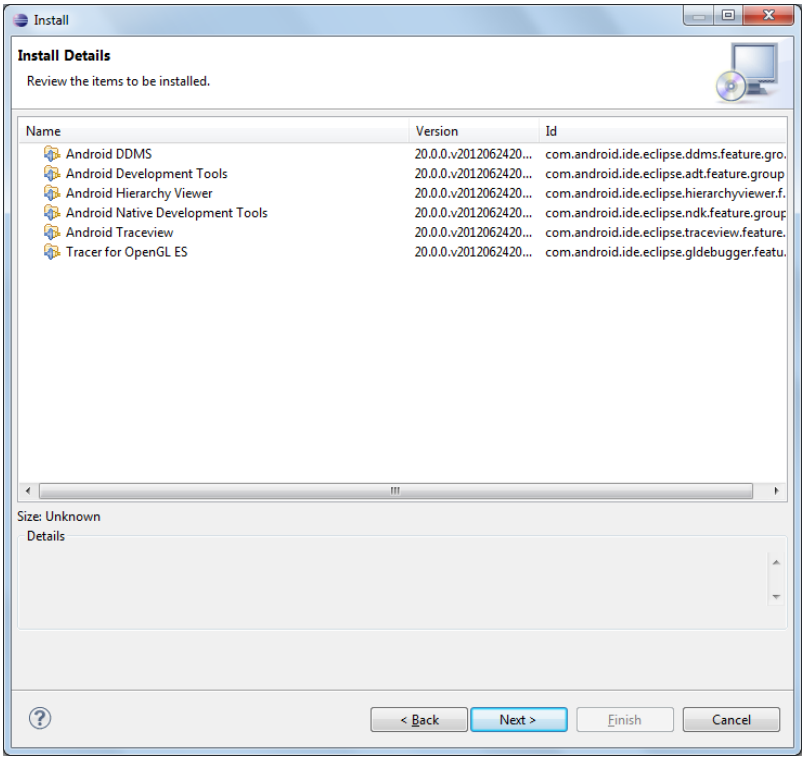
Caixa de Dialogo - Install

Agora marque as opções “Developer Tools” e “NDK Plugin” conforme mostra a figura seguinte:



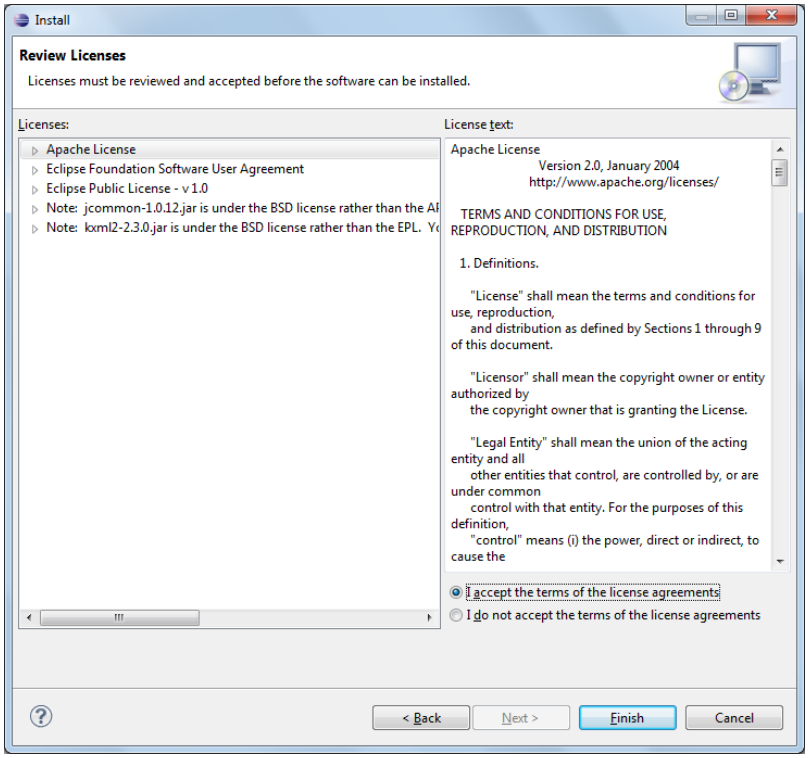
Caixa de Dialogo - Install

Após fazer isso clique no botão “Next”, e em seguida será mostrada a próxima tela, conforme demonstra a figura seguinte:



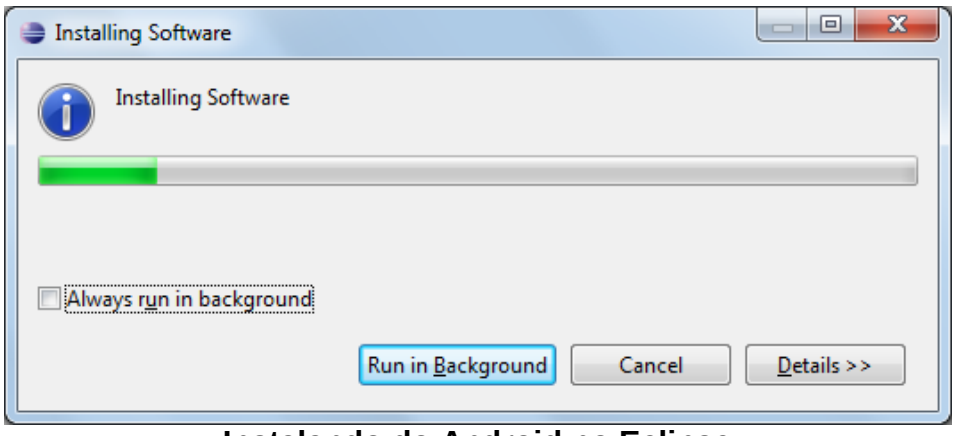
Caixa de Dialogo - Install

Ao clicar em “Next” será aberta uma tela de “Licença”, conforme a figura abaixo :


Install
Review Licenses
Licenses must be reviewed and accepted before the software can be installed.
Licenses:
License text:
Apache License
Eclipse Foundation Software User Agreement
Eclipse Public License - v 1.0
Note: jcommon-1.0.12.jar is under the BSD license rather than the AP
Note: lxml2-2.3.0.jar is under the BSD license rather than the EPL. Yo
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.
"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the
I accept the terms of the license agreements
I do not accept the terms of the license agreements
< Back
Next >
Finish
Cancel

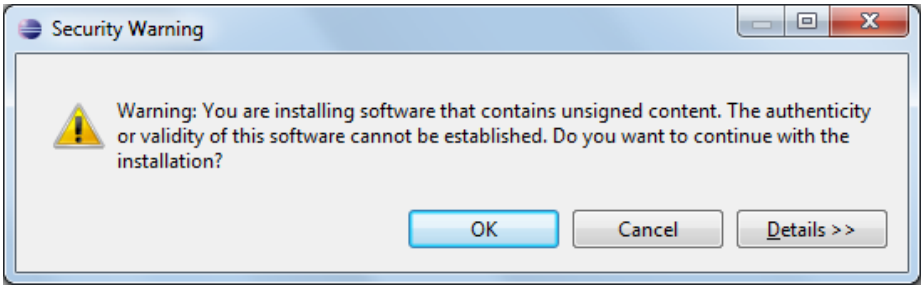
Caixa de Dialogo - Install

Para prosseguir com a instalação marque a opção “I accept the terms of the license agreements” e em seguida, clique no botão “Finish”. Após isso ocorrerá alguns processos, como demonstra a figura seguinte, aguarde até terminar.



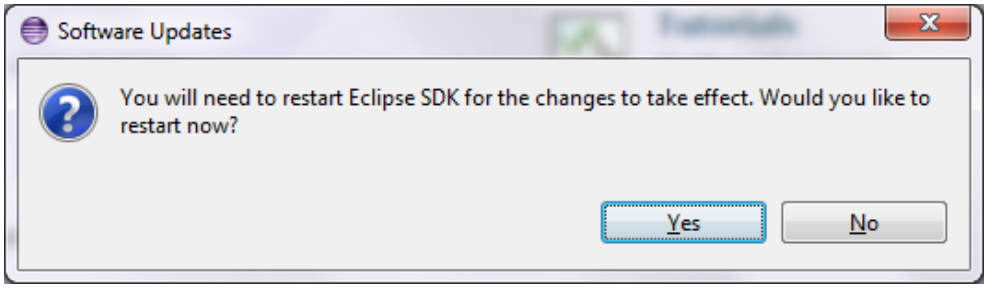
Instalando do Android no Eclipse

Se em algum momento durante o processo for exibida uma caixa de diálogo, conforme mostra a figura seguinte:


Security Warning
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?
OK
Cancel
Details >>

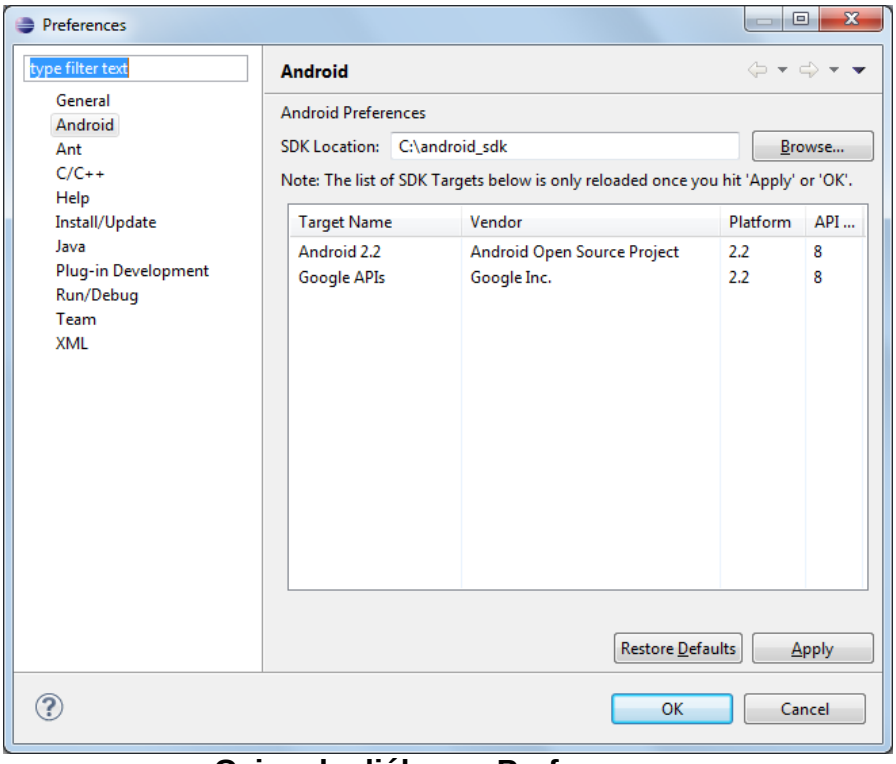
Caixa de mensagem

Pode clicar em “OK” sem problemas, e o processo se completará. Após o término do processo você deve reiniciar o Eclipse, clicando em “Yes”, na mensagem abaixo:



Caixa de mensagem

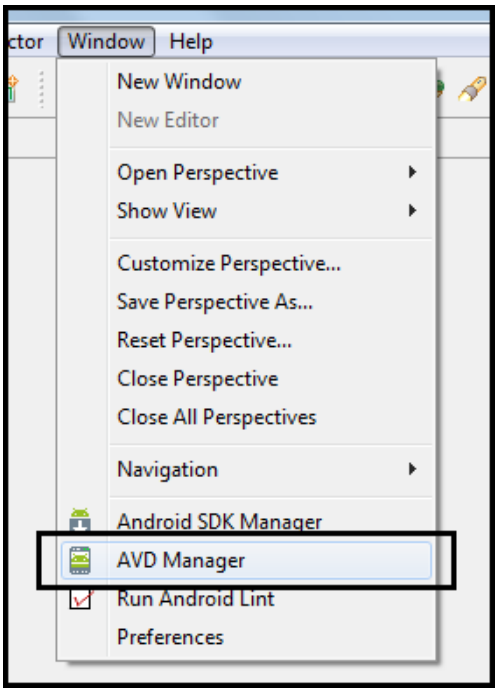
Após o Eclipse ter reiniciado, vamos fazer agora as configurações para fazer conexão com o emulador do Android. Vamos no menu “Window” / “Preferences”. Aberta a caixa de diálogo, selecione o item “Android” e será mostrada uma tela, conforme demonstra a figura seguinte:



Caixa de diálogo - Preferences

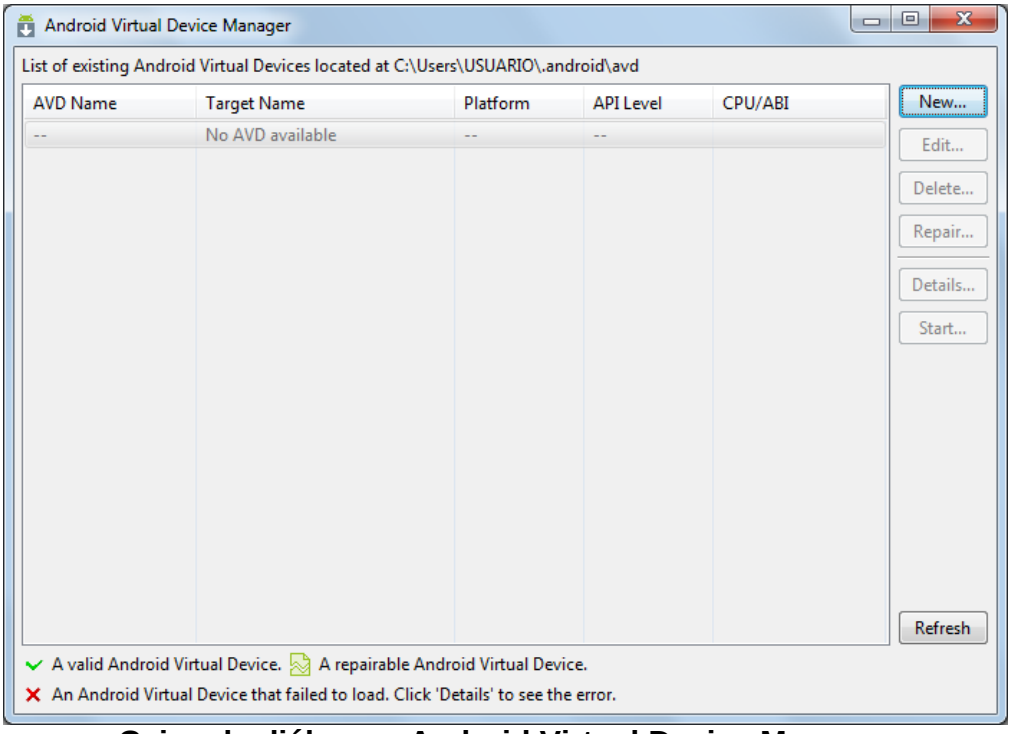
Certifique-se de que os dados estão de acordo com a figura acima (apontando para o diretório “C:\android_sdk\”). Caso não esteja, especifique o diretório onde você instalou o Android SDK (no caso o “C:\android_sdk\”).

Para finalizar vamos definir um dispositivo virtual, conhecido como AVD (Android Virtual Device), onde nossas aplicações daqui para frente serão executadas. Para isso, vá no menu “Windows” / “AVD Manager”, conforme mostra a figura seguinte:



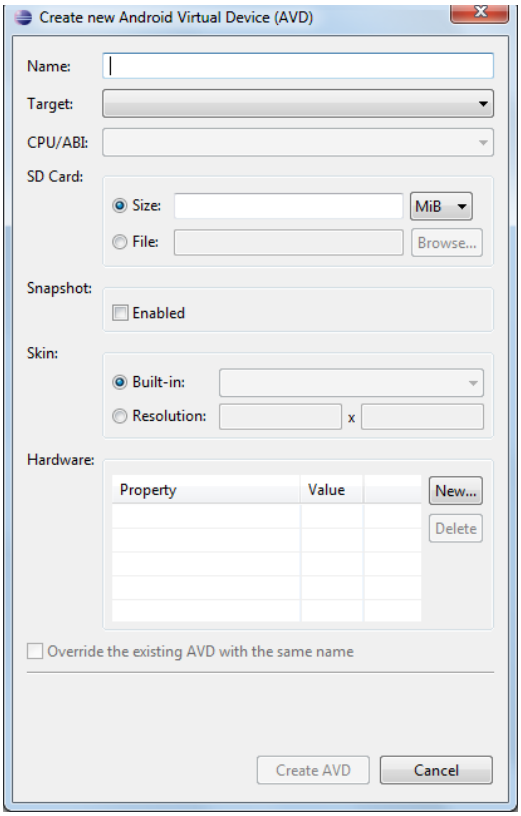
AVD Manager

Feito o procedimento acima, será aberta uma tela conforme mostra a figura seguinte:



Caixa de diálogo – Android Virtual Device Manager

Para criarmos um dispositivo virtual clique no botão “New”, e em seguida será aberta uma tela conforme mostra a figura seguinte:



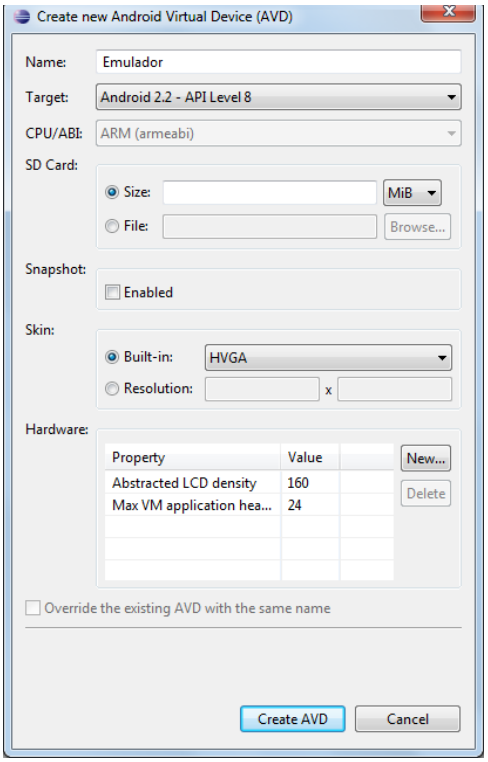
Caixa de diálogo - Crete new AVD

Inicialmente, vamos configurar o básico pra executarmos a nossa aplicação. Em “Name” você define o nome do AVD, vamos chamá-lo de “Emulador”.

Em “Target” definirmos a plataforma-alvo a ser executada, neste caso só temos uma, o “Android 2.2 - API Level 8”. Vamos selecioná-la.

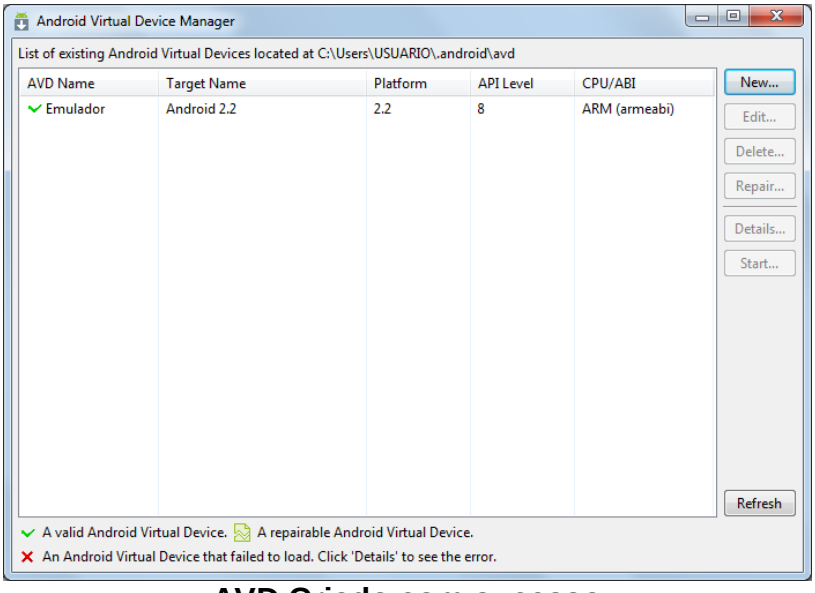
Em Skin na opção “Built-in” a opção padrão é “Default (WVGA800)”. Vamos mudar essa opção para “HVGA”.

Depois de preencher todos os campos, a tela de criação do AVD deve estar de acordo com a figura abaixo:



Caixa de diálogo - Create new AVD

Para criarmos nosso AVD, clique no botão “Create AVD” e pronto. O resultado você confere na figura seguinte:



AVD Criado com sucesso

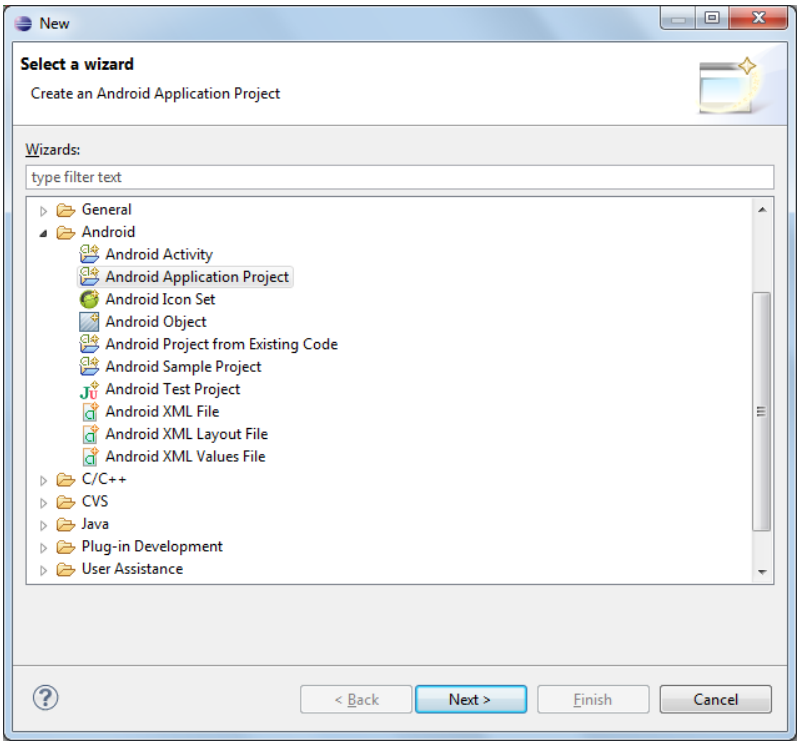
Até aqui aprendemos a instalar e configurar a plataforma Android no Eclipse. Agora vamos dar nossos primeiros passos desenvolvendo uma aplicação básica no Android.

Capitulo 4 Construindo nossas aplicações no Android

Vamos colocar a mão na massa ? A partir de agora iremos começar a desenvolver as nossas aplicações no Android utilizando os componentes descritos no capítulo anterior. Começaremos com aplicações simples e aos poucos iremos evoluir, criando aplicações mais ricas.

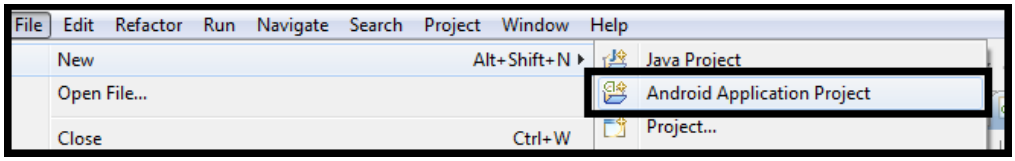
4.1) Desenvolvendo uma Calculadora Básica

Vamos construir a nossa primeira aplicação que vai consistir em uma calculadora básica com as quatro operações aritméticas. Para criar um projeto em Android (conforme já foi mostrado mas, mostro novamente aqui) vamos no menu “File”/”New” e em seguida selecione “Others”. Em seguida “expanda” a pasta “Android” e logo após selecione a opção “Android Application Project”, conforme mostra a figura seguinte :



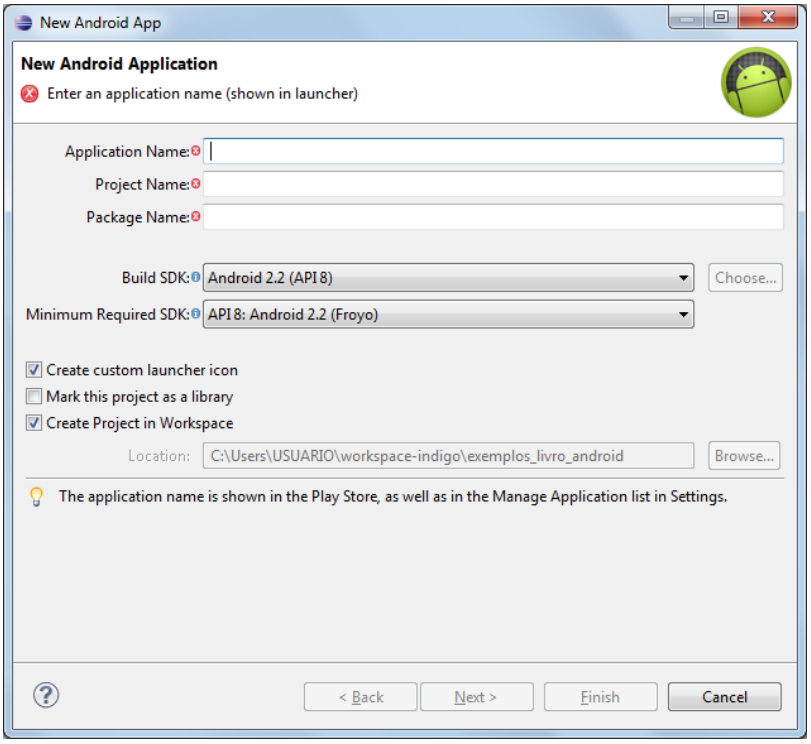
Android Application Project

OBS : Possivelmente (isso depende das configurações do Eclipse), você poderá ver o subitem “Android Application Project” no próprio menu “File” / “New”, conforme demonstra a imagem seguinte. Caso isso aconteça, você pode selecionar que é o mesmo processo dos passos descritos acima.



Android Application Project (pelo menu)

Seguido um dos passos descritos acima, irá se abrir a caixa de diálogo abaixo:



Criando o projeto Calculadora

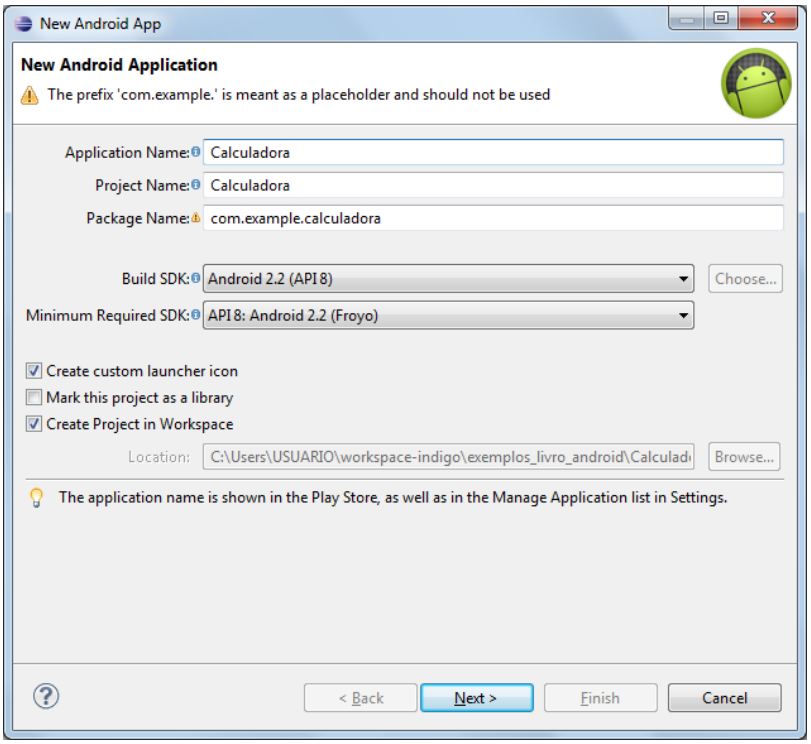
Agora vamos preencher os campos, conforme abaixo:

Application Name : Calculadora

Project Name : Calculadora

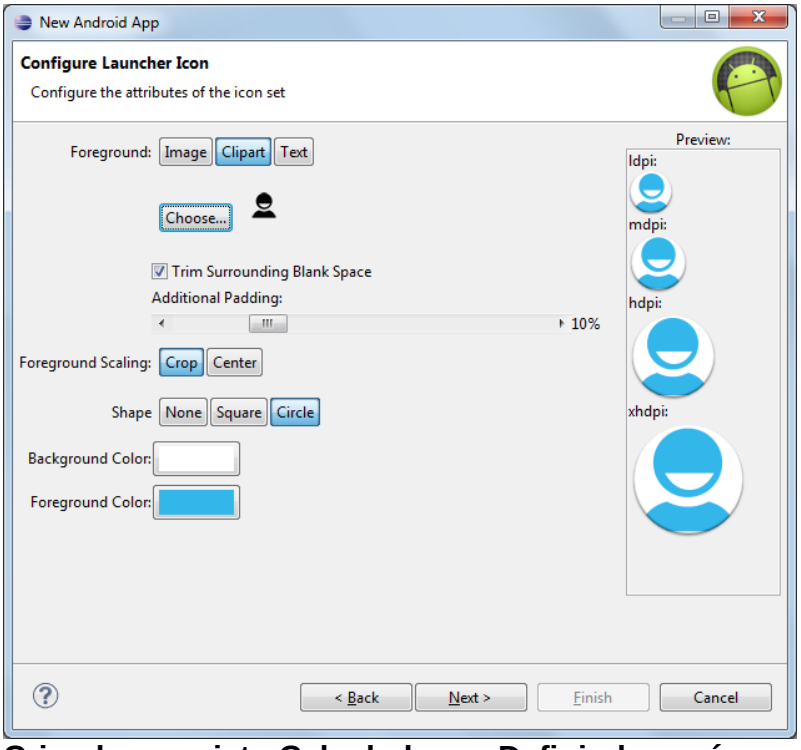
Package Name : com.example.calculadora

Confira como ficou na figura abaixo:


New Android App
New Android Application
The prefix 'com.example.' is meant as a placeholder and should not be used
Application Name: Calculadora
Project Name: Calculadora
Package Name: com.example.calculadora
Build SDK: Android 2.2 (API 8)
Choose...
Minimum Required SDK: API 8: Android 2.2 (Froyo)
Create custom launcher icon
Mark this project as a library
Create Project in Workspace
Location: C:\Users\USUARIO\workspace-indigo\exemplos_livro_android\Calculad
Browse...
The application name is shown in the Play Store, as well as in the Manage Application list in Settings.
< Back
Next >
Finish
Cancel

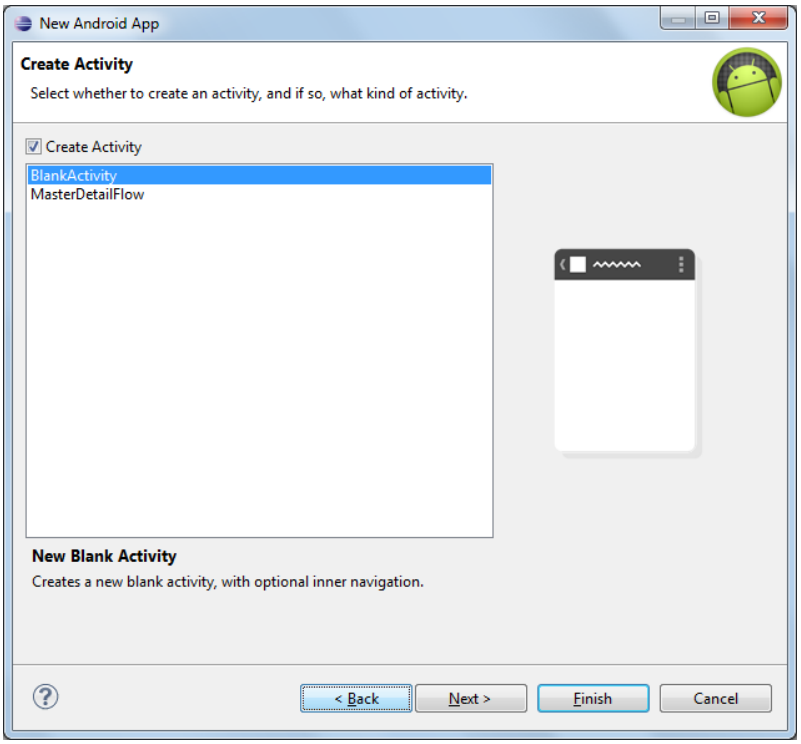
Criando o projeto Calculadora – Campos preenchidos

Agora na próxima seção (clicando em "Next") escolhemos o ícone para a nossa aplicação (conforme demonstra a imagem seguinte). Fique a vontade para escolher seu ícone (Dica : escolha ícones que tenham de uma certa forma, relação com a aplicação em desenvolvimento).



Criando o projeto Calculadora – Definindo um ícone

Depois de definir seu ícone vamos para a próxima etapa, onde vamos escolher qual tipo de Activity iremos criar (por padrão, o BlackActivity), conforme demonstra a próxima imagem:



Criando o projeto Calculadora – Definindo a Activity

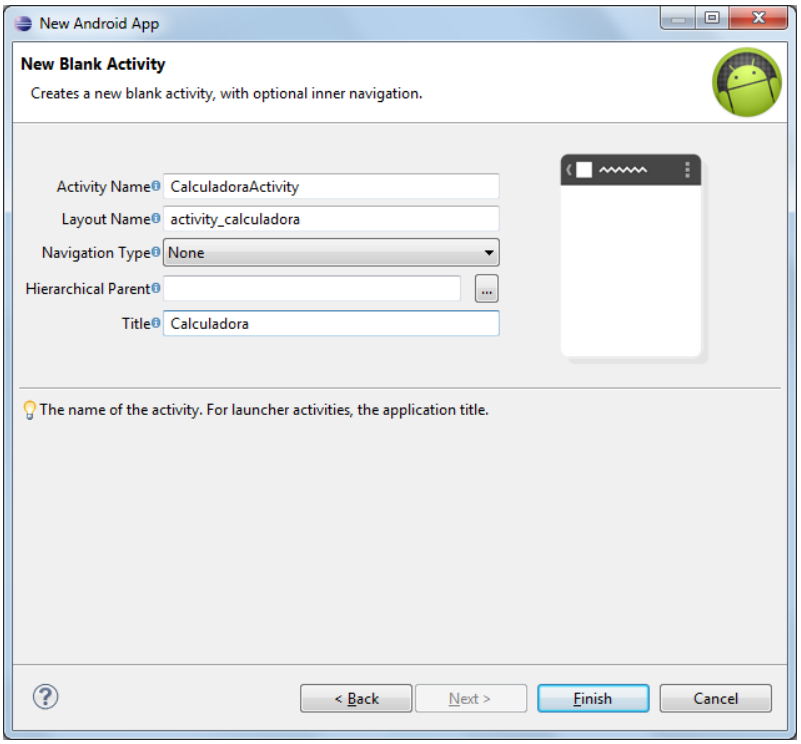
Agora na próxima seção (clicando em “Next”) vamos preencher as informações da Activity, conforme é mostrado abaixo:

Activity Name : CalculadoraActivity

Layout Name : activity_calculadora

Title : Calculadora

Confira como ficou na figura seguinte:



Criando o projeto Calculadora – Informações preenchidas

Depois de preenchidas as informações, vamos criar o nosso projeto clicando no botão “Finish”. Feito isso o nosso projeto será criado.

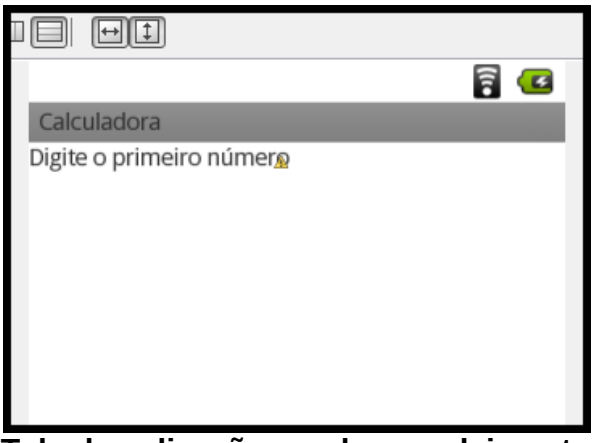
A primeira coisa que iremos fazer a alterar a estrutura de layout (trocar de **RelativeLayout** para **LinearLayout**) que vai “comportar” nossos componentes.

Depois de alterar a estrutura de layout vamos selecionar o componente **TextView** na tela (cuja frase está escrito “Hello World”) e vamos alterar as seguintes propriedades, como segue:

TextView

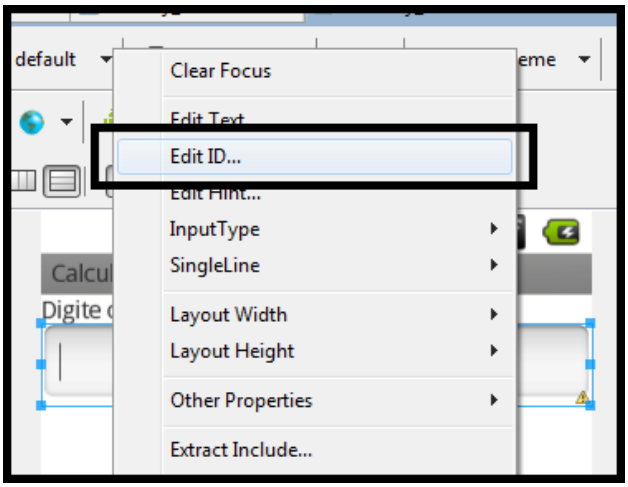
Propriedade	Valor
Padding	(deixar em branco)
Text	Digite o primeiro número

Veja o resultado:



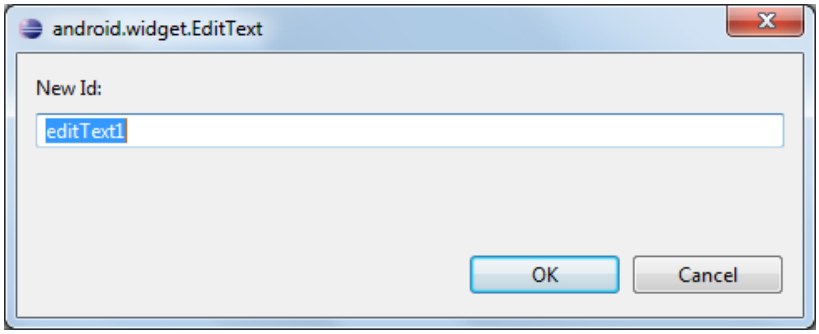
Tela da aplicação em desenvolvimento

Agora arraste e solte um componente “Plain Text” (**EditText**) abaixo do título e em seguida vamos atribuir um nome de componente para ele (por padrão, ele coloca “editText1”), clicando com o botão direito do mouse sobre ele e em seguida, selecionar “Edit ID”. Confira na imagem abaixo:



Alterando o nome do componente

Feito isso, vai ser aberto uma caixa de diálogo conforme mostra a imagem seguinte:



Caixa para alteração do nome do componente

Conforme falei, o nome do componente é “editText1”. Agora vamos mudar o nome desse componente para “ednumero1” (sem aspas, é claro). Feito isso vamos clicar em “OK” para confirmar a alteração.
Porque alterar a sua ID ? Isso é necessário pois vamos “manipular” esse componente através do código Java, então nada mais justo do que trabalhar com componentes cujos nomes estejam de forma clara e organizada.

Agora arraste e solte um componente **TextView** abaixo da caixa de texto que inserimos, e em seguida altere as seguintes propriedades:

TextView

Propriedade	Valor
Text	Digite o segundo número

Logo após , arraste e solte um componente Plain Text (**EditText**) abaixo do componente acima inserido, e altere seu nome (ID) para “ednumero2” (conforme já foi mostrado).

Veja o resultado:

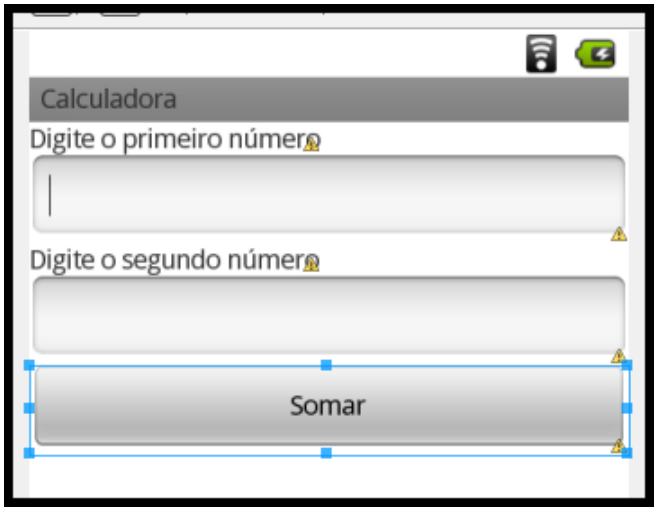
Tela da aplicação em desenvolvimento

Agora vamos adicionar um componente **Button** abaixo da caixa de texto, que vai ser o nosso botão de “somar” os números. Depois de adicionar, vamos alterar as suas propriedades, conforme é mostrado abaixo:

Button

Propriedade	Valor
Text	Somar
Width	fill_parent

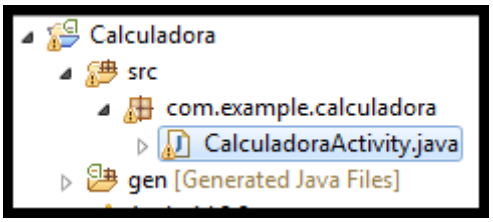
Depois disso vamos atribuir um nome (ID) para o componente, que vai se chamar “btsomar”. Veja o resultado abaixo:



Tela da aplicação em desenvolvimento

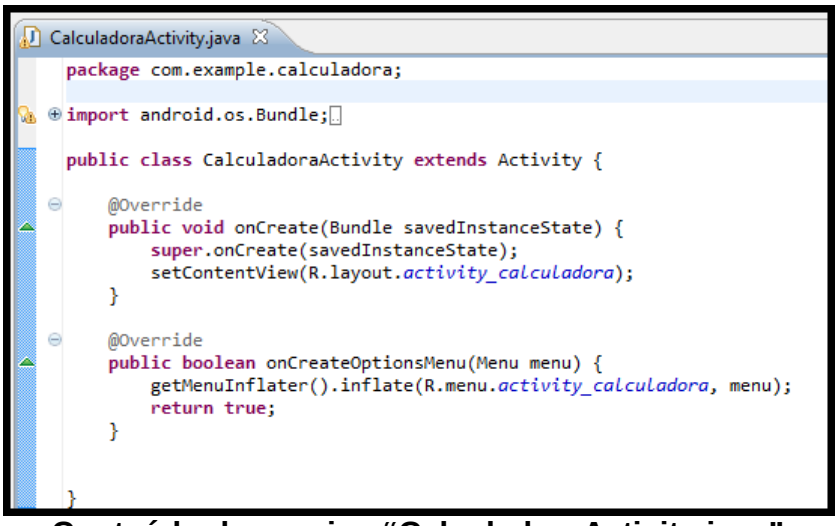
Para começarmos, vamos fazer o teste da nossa aplicação realizando somente soma dos números (implementaremos as outras operações restantes daqui a pouco). Agora salve o arquivo “activity_calculadora.xml” , para confirmar todas as alterações feitas, antes de trabalharmos com a parte da programação Java (que vai fazer uso dos componentes da tela da aplicação via código).

Depois de salvar o arquivo XML vamos abrir o arquivo “CalculadoraActivity.java” (situado no pacote “com.example.calculadora”, que fica dentro do diretório “src”). Veja a imagem abaixo:



Arquivo “CalculadoraActivity.java”

Feito isso será aberto o seu conteúdo conforme é demonstrado na imagem seguinte:

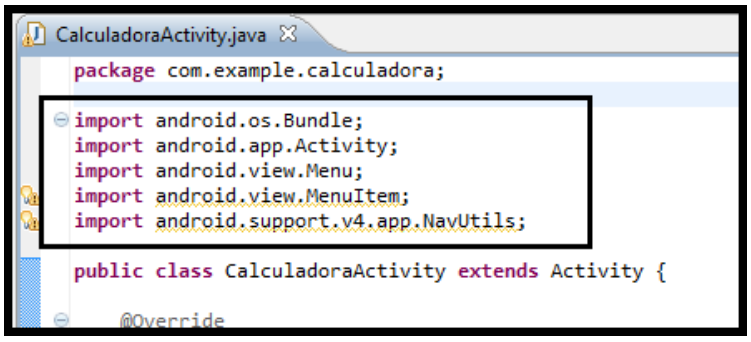


Conteúdo do arquivo "CalculadoraActivity.java"

Se você observar no código acima, na seção onde se declaram os pacotes, existe a seguinte instrução :

```
import android.os.Bunble;
```

Nessa linha se você observar (conforme demonstra a figura acima), existe um sinal de "+", que na verdade indica que há mais de uma importação (processo esse que o eclipse faz para "simplificar" e "organizar" a compreensão do código). Para você visualizar todos os pacotes utilizados basta clicar nesse sinal. Confira o resultado na próxima figura:



Visualizando todos os pacotes

Para começar, vamos importar alguns pacotes da plataforma Android que serão necessários para o desenvolvimento da nossa aplicação. Na seção onde se encontram os pacotes importados , vamos importar mais alguns pacotes digitando as seguintes linhas de comando abaixo:

```
import android.widget.*;
import android.view.*;
import android.app.*;
```

Agora no código do nosso programa, antes da linha:

```
@Override
```

Digite:

```
EditText ednumero1,ednumero2;
Button btsomar;
```

Agora vamos à explicação do código acima. Como você pode ver , os widgets também podem ser usados no nosso código Java. Se no código XML eu possuir um widget do tipo **EditText**, para acessar esse componente pelo Java é preciso fazer uso da classe **EditText**. Cada widget no XML possui o seu respectivo "em classe" Java, logo, se possui um widget **Button**, para acessá-lo devo fazer uso da classe **Button** e assim vai.

Agora dentro do método **onCreate** após a linha:

```
setContentView(R.layout.activity_calculadora);
```

Digite as seguintes linhas de código:

```
ednumero1 = (EditText) findViewById(R.id.ednumero1);
ednumero2 = (EditText) findViewById(R.id.ednumero2);
btsomar  = (Button) findViewById(R.id.btsomar);
```

Agora vou explicar as linhas de comando acima que adicionamos. A linha:

```
ednumero1 = (EditText) findViewById(R.id.numero1);
```

Faz referência ao primeiro **EditText**, através do método **findViewById** com o parâmetro *"R.id.numero1"*.

Se lembra do nome da primeira **EditText** que está no código XML? Ela se chama *"ednumero1"*.

Vamos entender. Observe que para fazer referência ao **EditText** pelo método **findViewById** eu passei o parâmetro "*R.id.numero1"*.

Na segunda instrução que digitamos, para fazer referência à segunda **EditText**, cujo nome é "*ednumero2"*, pelo método **findViewById**, passei o parâmetro "*R.id.numero2"*.

Como você pode ver, estou fazendo uso da classe **R** (situado dentro do diretório "gen", presente no pacote "com.example.calculadora") que funciona

como interface entre o código Java e o arquivo XML. O procedimento é o mesmo para o componente **Button**.

Agora iremos adicionar um evento em nosso componente **Button** que será responsável por "detectar" toda vez que ele for "clicado", executando um conjunto de instruções após o evento (que vai consistir na soma dos números e na exibição do resultado). Para adicionarmos esse evento em nosso componente, basta escrevermos, após a última instrução que adicionamos, a seguinte linha de código:

```
btsomar.setOnClickListener(new View.OnClickListener() {

  @Override
  public void onClick(View arg0) {

      double num1 = Double.parseDouble(
      ednumero1.getText().toString());


      double num2 = Double.parseDouble(
      ednumero2.getText().toString());
      double soma = num1 + num2;

      AlertDialog.Builder dialogo = new
       AlertDialog.Builder(CalculadoraActivity.this);

      dialogo.setTitle("Resultado soma");

      dialogo.setMessage("A soma é " + soma);

      dialogo.setNeutralButton("OK", null);

      dialogo.show();

  }
});
```

Toda vez que eu clicar no botão ele irá mostrar o resultado da soma na tela através de uma caixa de mensagem. Ótimo!

Vamos executar a nossa aplicação? Para executar faça os mesmos procedimentos que já mostrei. O resultado da execução dessa aplicação você vê na figura seguinte:

Aplicação em execução

OBS: Provavelmente durante a execução da aplicação ao entrar com um número, deve ter surgido no dispositivo um teclado virtual (como mostra a figura acima), para ocultar ele é só pressionar ESC.

Irei descrever o código do evento de "clique". O método **setOnClickLisneter** serve para definir um evento de "clique" em um componente. Como parâmetro, criamos uma instância da interface **OnClickListener**, e dentro da mesma existe um método chamado **onClick**, que será disparado toda vez que o botão for clicado.

A linha:

```
double num1 =  Double.parseDouble(ednumero1.getText().toString());
```

Cria uma variável chamada *num1* e atribui a ela o valor que está contido dentro do componente identificado como *ednumero1*. Eu faço uso do método **parseDouble** da classe **Double** pois o conteúdo é uma **String**. Observem que chamo o método **getText** de *ednumero1* para retornar o conteúdo. Diferente de muitos métodos de retorno **String**, esse método **getText** não retorna uma **String**, mais sim um tipo chamado **Editable**. Por isso chamei o método **toString** de **getText** para que me retornasse uma **String**. A descrição da próxima linha e a similar ao que já foi explicado.

Logo após a soma dos números que será armazenada na variável *soma* , vem o código em seguida:

```
AlertDialog.Builder dialogo = new
AlertDialog.Builder(CalculadoraActivity.this);

dialogo.setTitle("Resultado soma");

dialogo.setMessage("A soma é " + soma);

dialogo.setNeutralButton("OK", null);

dialogo.show();
```

Que mostra a soma dos números digitados na tela. Para conseguirmos exibir uma mensagem na tela, tivemos que fazer uso da classe **AlertDialog.Builder**, responsável por criar caixas de diálogo e exibi-las. Vamos aos comentários.
A linha de comando:

```
AlertDialog.Builder dialogo = new
AlertDialog.Builder(CalculadoraActivity.this);
```

Cria a instância da classe **AlertDialog.Builder** que será representada e guardada dentro da variável *dialogo*. Na linha seguinte:

```
dialogo.setTitle("Resultado soma");
```

Define o título da caixa de diálogo através do método **setTitle**. Na linha seguinte:

```
dialogo.setMessage("A soma é " + soma);
```

Define a mensagem a ser exibida através do método **setMessage**. Na linha seguinte:

```
dialogo.setNeutralButton("OK", null);
```

Define o botão “OK” da caixa de texto através do método **setNeutralButton**. O parâmetro **null** indica que nenhuma ação será executada quando o botão for clicado (simplesmente a caixa será fechada e nada mais). E para finalizar:

```
dialogo.show();
```

Que é responsável por “exibir” a mensagem na tela por imediato.

Agora vamos continuar as outras operações certo ? Retornaremos então para a tela da nossa aplicação e vamos adicionar mais 3 botões referentes as operações restantes. Vamos adicionar na tela mais três botões como segue (um em baixo do outro, conforme a sequência abaixo):

Button

Propriedade	Valor
Id	btsubtrair
Text	Subtrair
Width	fill_parent

Button

Propriedade	Valor
Id	btmultiplicar
Text	Multiplicar
Width	fill_parent

Button

Propriedade	Valor
Id	btdividir
Text	Dividir
Width	fill_parent

Depois de “finalizado” o que foi se pedido acima, veja como ficou a tela da nossa aplicação:

Tela da aplicação da calculadora

Agora retornando para o código do arquivo “CalculadoraActivity.java”, vamos declarar mais três atributos (variáveis) que vão corresponder aos botões que representam as operações restantes, conforme destaca a linha em negrito:

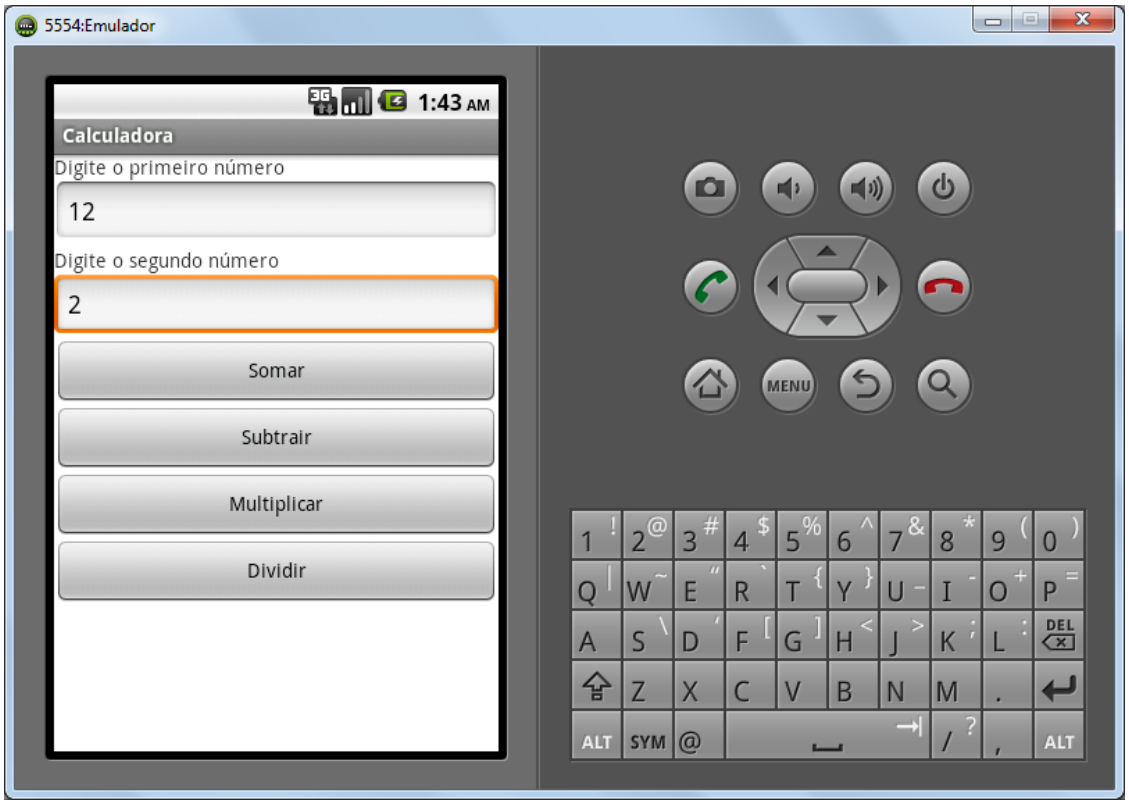
```
:
Button btsomar, btsubtrair,btmultiplicar, btdividir;
:
```

Agora vamos atribuir para cada botão um evento de clique, fazendo com que eles efetuem a sua respectiva operação aritmética. Vamos continuar a codificação do método onCreate , digitando o seguinte código abaixo:

```
btsubtrair = (Button) findViewById(R.id.btsubtrair);

btmultiplicar=(Button)findViewById(R.id.btmultiplicar);

btdividir = (Button) findViewById(R.id.btdividir);

btsubtrair.setOnClickListener(new View.OnClickListener() {

   @Override
   public void onClick(View arg0) {

     double num1 = Double.parseDouble
     (ednumero1.getText().toString());

     double num2 = Double.parseDouble
     (ednumero2.getText().toString());

      double soma = num1 - num2;

      AlertDialog.Builder dialogo = new
      AlertDialog.Builder(CalculadoraActivity.this);

      dialogo.setTitle("Resultado subtração");

      dialogo.setMessage("A subtração é " + soma);

      dialogo.setNeutralButton("OK", null);

      dialogo.show();

    }
   });


 btmultiplicar.setOnClickListener(new View.
 OnClickListener() {

   @Override
    public void onClick(View arg0) {

        double num1 = Double.parseDouble
        (ednumero1.getText().toString());
```

```
        double num2 = Double.parseDouble
        (ednumero2.getText().toString());

        double soma = num1 * num2;

         AlertDialog.Builder dialogo = new
         AlertDialog.Builder(CalculadoraActivity.this);

        dialogo.setTitle("Resultado multiplicação");

        dialogo.setMessage("A multiplicação é " + soma);

        dialogo.setNeutralButton("OK", null);

        dialogo.show();

      }
});

btdividir.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {

       double num1 = Double.parseDouble
       (ednumero1.getText().toString());

      double num2 = Double.parseDouble
      (ednumero2.getText().toString());

      double soma = num1 / num2;

      AlertDialog.Builder dialogo = new
      AlertDialog.Builder(CalculadoraActivity.this);

      dialogo.setTitle("Resultado divisão");

      dialogo.setMessage("A divisão é " + soma);

      dialogo.setNeutralButton("OK", null);

      dialogo.show();

 } });
```

Depois de escrever o código acima, salve o arquivo e em seguida teste a aplicação. Veja o resultado na figura seguinte:



Aplicação da calculadora em execução

4.2) Desenvolvendo uma aplicação simples de compras

Agora para aprimorarmos o nosso conhecimento no desenvolvimento de aplicações para Android, vamos criar um outro aplicativo que consiste em um sistema de compras, bem simples. Em nossa aplicação terei disponível cinco produtos: Arroz (R$ 2,69) , Leite (R$ 5,00) , Carne (R$ 10,00), Feijão (R$ 2,30) e Refrigerante Coca-Cola (R$ 2,00). Nessa aplicação eu marco os itens que quero comprar e no final o sistema mostra o valor total das compras.

Na aplicação que iremos desenvolver vamos utilizar os seguintes widgets : **TextView**, **CheckBox** e **Button**.

Bom, vamos criar um novo projeto no Eclipse para Android chamado “SistemaDeCompras”. Siga os dados do projeto abaixo:

Application Name: SistemaDeCompras

Project Name: SistemaDeCompras

Package Name : com.example.sistemadecompras

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: ComprasActivity

Layout Name : activity_compras

Title : Sistema de Compras

Depois de carregado e criado o projeto, vamos alterar a estrutura de layout padrão (**RelativeLayout**) para **LinearLayout**. Em seguida, modifique o componente **TextView** situado na tela, de acordo com a tabela abaixo:

TextView

Propriedade	Valor
Padding	(deixar em branco)
Text	Escolha seu produto

Feito o que se foi pedido, adicione os seguintes componentes na sequência:

CheckBox

Propriedade	Valor
Text	Arroz (R$ 2,69)
Id	chkarroz

CheckBox

Propriedade	Valor
Text	Leite (R$ 5,00)
Id	chkleite

CheckBox

Propriedade	Valor
Text	Carne (R$ 9,70)
Id	chkcarne

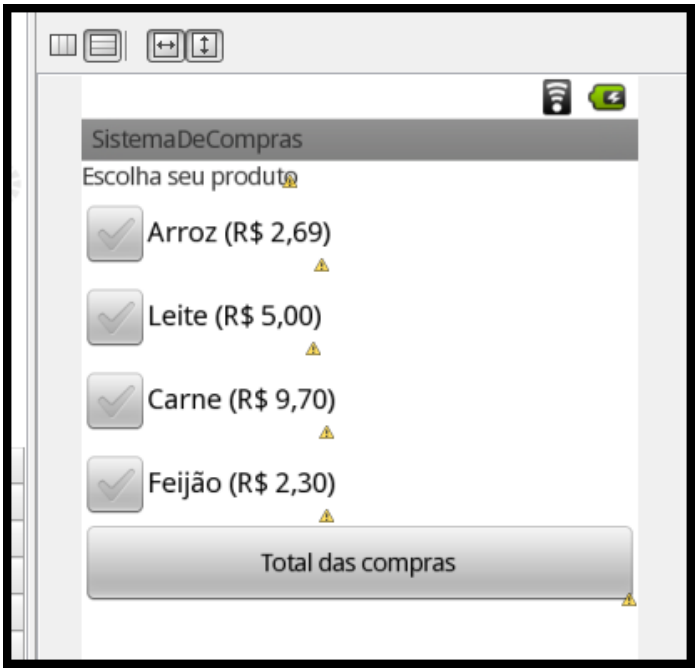
CheckBox

Propriedade	Valor
Text	Feijão (R$ 2,30)
Id	chkfeijao

Button

Propriedade	Valor
Text	Total das compras
Id	bttotal
Width	fill_parent

Ao final, o layout da nossa aplicação deve estar de acordo com a figura seguinte:



Layout da tela da aplicação

Agora vamos modificar o arquivo “ComprasActivity.java“. O código “completo” desse arquivo será como o código que é exibido abaixo:

```
package com.example.sistemadecompras;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.*;
import android.view.*;
import android.app.*;

public class ComprasActivity extends Activity {

      CheckBox chkarroz, chkleite, chkcarne, chkfeijao;

      Button bttotal;
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_compras);

    chkarroz = (CheckBox)
    findViewById(R.id.chkarroz);

    chkleite = (CheckBox)
    findViewById(R.id.chkleite);

    chkcarne = (CheckBox)
    findViewById(R.id.chkcarne);

    chkfeijao = (CheckBox)
    findViewById(R.id.chkfeijao);


    Button bttotal = (Button) findViewById(R.id.bttotal);

    bttotal.setOnClickListener(new View.OnClickListener() {

     @Override
     public void onClick(View arg0) {

       double total =0;

       if(chkarroz.isChecked())
         total += 2.69;

       if(chkleite.isChecked())
         total += 5.00;

       if(chkcarne.isChecked())
        total += 9.7;

       if(chkfeijao.isChecked())
         total += 2.30;

       AlertDialog.Builder dialogo = new
       AlertDialog.Builder(ComprasActivity.this);

       dialogo.setTitle("Aviso");
       dialogo.setMessage("Valor total da compra :" +
       String.valueOf(total));

       dialogo.setNeutralButton("OK", null);

       dialogo.show();


        }
  });

}
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.
        activity_compras, menu);
        return true;
    }


}
```

Agora vou descrever o código situado no método **onClick**. Dentro do método eu crio uma variável chamada *total* que armazena o valor total da compra. Observe que eu tenho quatro estruturas *if's* onde cada uma verifica se um determinado item foi marcado, se foi, incrementa o valor do item na variável *total*. No final é exibido o valor total das compras na tela.

Vamos roda nossa aplicação? O resultado você confere na figura seguinte:

Aplicação simples de compras em execução

4.3) Desenvolvendo uma aplicação de cálculo de salário

Agora vamos desenvolver uma nova aplicação que vai consistir em um sistema onde nós vamos digitar o salário de um funcionário permitindo escolher o seu percentual de aumento, que pode ser de 40% , 45% e 50%. Ao final de tudo, o sistema irá mostrar o salário reajustado com o novo aumento.

Para essa aplicação vamos utilizar os seguintes widgets : **TextView, EditText, RadioButton** e **Button**.

Bom, vamos lá! Crie um novo projeto Android com os seguintes dados abaixo:

Application Name: CalculoDeSalario

Project Name: CalculoDeSalario

Package Name : com.example.calculodesalario

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: SalarioActivity

Layout Name : activity_salario

Title : Calculo de salário

Depois de carregado e criado o projeto, vamos alterar a estrutura de layout padrão (**RelativeLayout**) para **LinearLayout**. Em seguida, modifique o componente **TextView** situado na tela, de acordo com a tabela abaixo:

TextView

Propriedade	Valor
Padding	(deixar em branco)
Text	Digite seu salário

Em seguida, adicione os seguintes componentes na sequência:

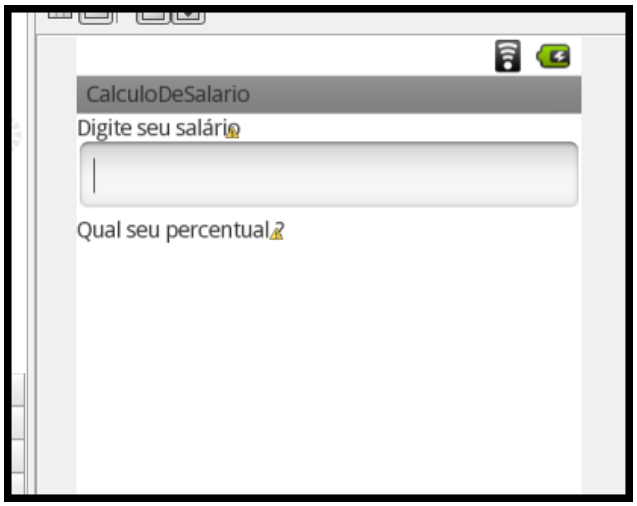
EditText (Number Decimal)

Propriedade	Valor
Text	
Id	edsalario
Width	fill_parent

TextView

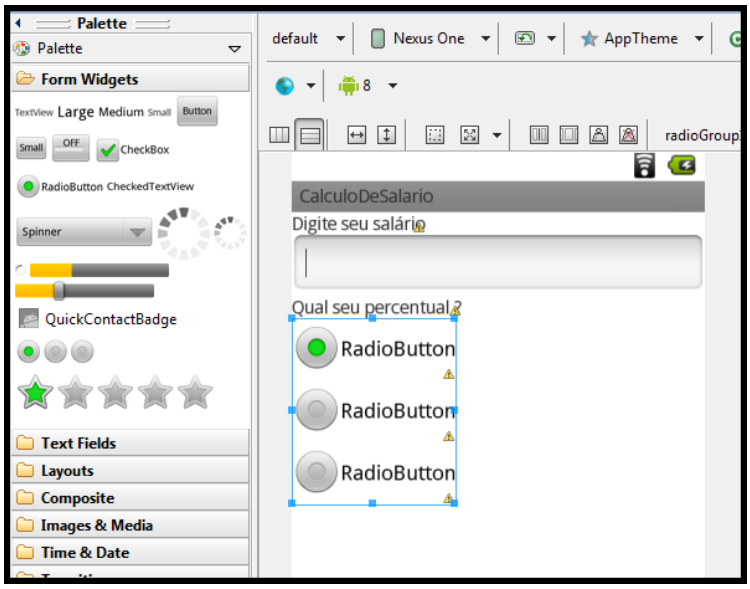
Propriedade	Valor
Text	Qual é o seu percentual ?

Seguindo os passos acima ate aqui, a aplicação deve estar de acordo com o dá figura abaixo:



Tela de layout da aplicação Cálculo de salário

Bom, agora vamos adicionar um componente, ou melhor, uma estrutura que será responsável por agrupar as RadioButtons dentro dela, que se chama **RadioGroup** (Para mais informações veja o Capítulo 4). O **RadioGroup** já oferece por padrão três RadioButtons, que é quantidade necessária para a nossa aplicação. Clique a arraste o componente abaixo do ultimo widget adicionado. O resultado você confere na figura abaixo:



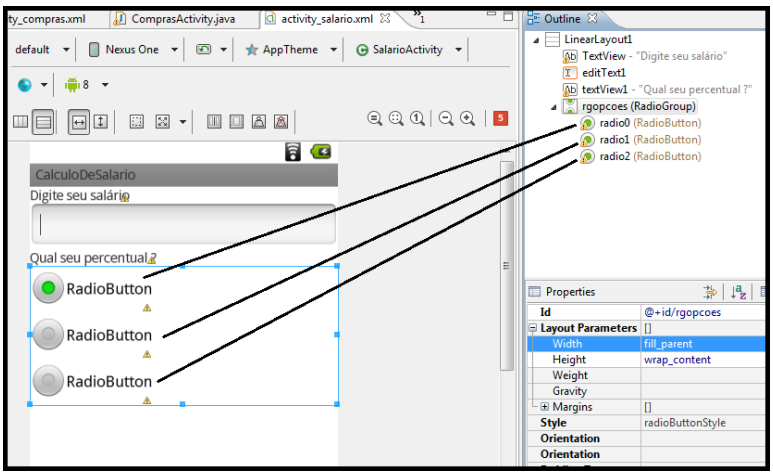
Estrutura RadioGroup inserida

Com o **RadioGroup** selecionado, modifique as propriedades abaixo:

RadioGroup

Propriedade	Valor
Width	fill_parent
Id	rgopcoes

Observe que dentro do **RadioGroup** existem três elementos, cada um deles do tipo **RadioButton** e identificados por um nome. Se você observar no Eclipse, à direita da tela da aplicação, existe uma seção chamada “Outline”, onde nela posso visualizar toda a estrutura dos componentes que estão na minha aplicação. Confira na figura abaixo:



Guia “Outline”

Agora modifique as propriedades das RadioButons de acordo com as indicações abaixo:

radio0

Propriedade	Valor
Text	40%
Id	rb40

radio1

Propriedade	Valor
Text	45%
Id	rb45

radio2

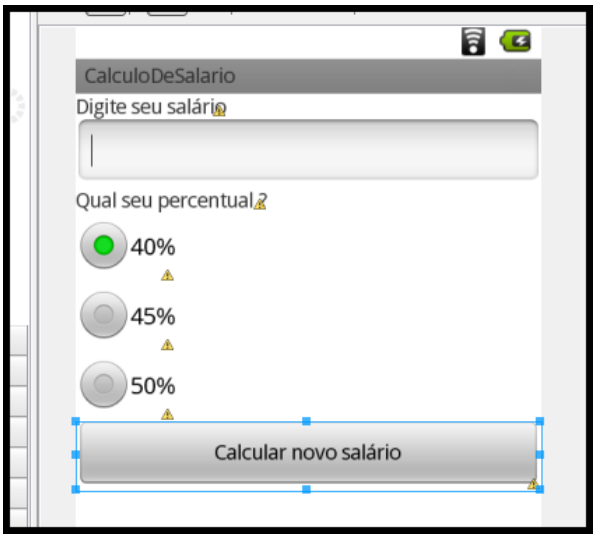
Propriedade	Valor
Text	50%
Id	rb50

Agora, vamos adicionar uma **Button**, simplesmente clicando e arrastando o componente na tela. Agora um detalhe, é para colocar esse componente na tela do dispositivo mas FORA da área do **RadioGroup**.

Depois de colocar o **Button**, modifique as propriedades abaixo:

Propriedade	Valor
Text	Calcular novo salário
Id	btcalcular
Width	fill_parent

Depois de inserir todos os componentes citados, o layout da aplicação deve ficar de acordo com a figura seguinte:



Layout da tela da aplicação

Vamos analisar agora parte de um trecho de código produzido. Como havia falado acima, as **RadioButtons** precisam ficar dentro de uma estrutura chamada **RadioGroup** certo ? Vamos ver como isso é estruturado dentro de um código XML, como você confere abaixo:

```
<RadioGroup
        android:id="@+id/rgopcoes"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >

        <RadioButton
            android:id="@+id/rb40"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="40%" />

        <RadioButton
            android:id="@+id/rb45"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="45%" />

        <RadioButton
            android:id="@+id/rb50"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="50%" />

    </RadioGroup>
```

Observe acima que logo após a definição da estrutura **RadioGroup**, existe dentro dela as RadioButtons, que serão utilizadas na aplicação.

No arquivo “SalarioActivity.java” vamos colocar o seguinte código abaixo:

```
package com.example.calculodesalario;

import android.os.Bundle;
import android.app.Activity;
import android.widget.*;
import android.view.*;
import android.app.*;

public class SalarioActivity extends Activity {

  RadioGroup rgopcoes;
  Button btcalcular;

  EditText edsalario;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_salario);

    edsalario = (EditText)findViewById
    (R.id.edsalario);

    rgopcoes = (RadioGroup) findViewById
    (R.id.rgopcoes);

    btcalcular = (Button) findViewById(R.id.btcalcular);

    btcalcular.setOnClickListener(new View.
    OnClickListener() {

      @Override
      public void onClick(View arg0) {

      double salario = Double.parseDouble
      (edsalario.getText().toString());

      int op = rgopcoes.getCheckedRadioButtonId();

      double novo_salario = 0;

      if(op==R.id.rb40)
          novo_salario = salario + (salario * 0.4);
      else if(op==R.id.rb45)
          novo_salario = salario + (salario * 0.45);
      else
          novo_salario = salario + (salario * 0.5);


      AlertDialog.Builder dialogo = new
      AlertDialog.Builder(SalarioActivity.this);

      dialogo.setTitle("Novo salário");

      dialogo.setMessage("Seu novo salário é : R$" +
      String.valueOf(novo_salario));
```

```
        dialogo.setNeutralButton("OK", null);

        dialogo.show();


                    }
        });
    }


}
```

Vamos à explicação de alguns códigos interessantes. Dentro do método **onClick**, eu realizo o cálculo do novo salário do funcionário. Os primeiros códigos do evento são similares de programas anteriores que já foram devidamente explicados. A linha:

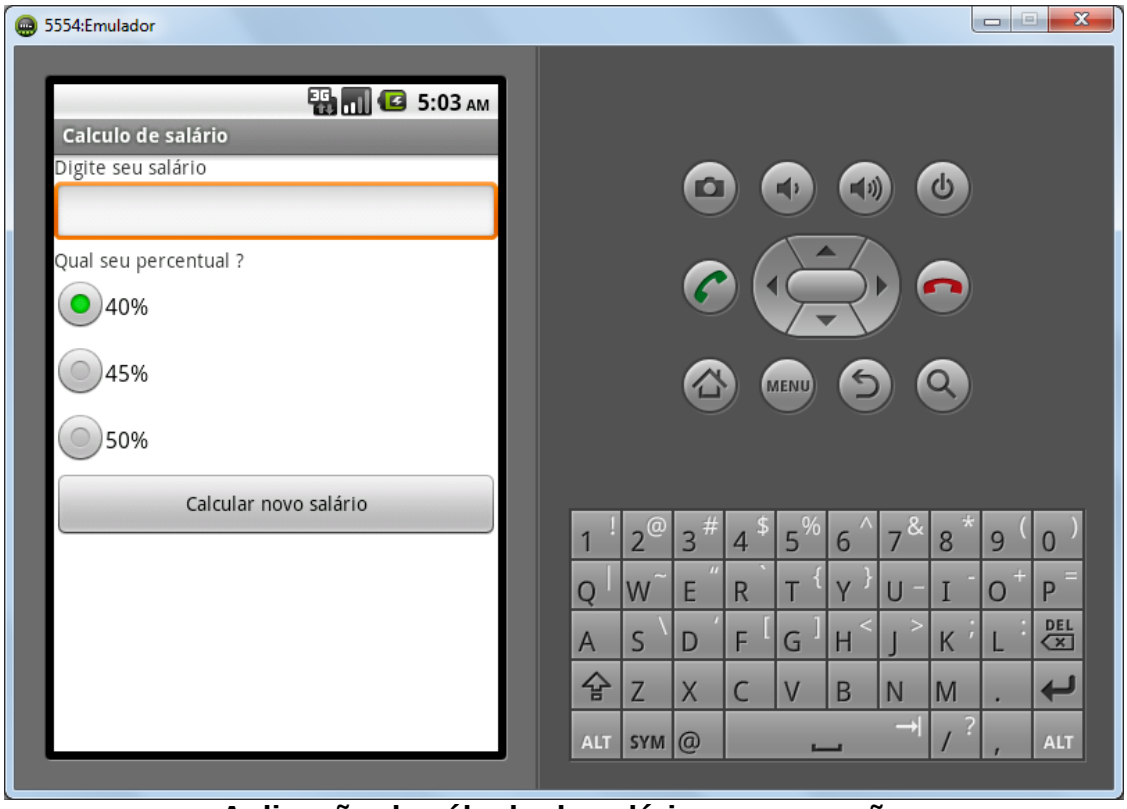
```
int op = rg.getCheckedRadioButtonId();
```

Cria uma variável *op* e retorna para ela o Id da opção selecionada, ou seja, qual **RadioButton** foi selecionada.

Agora na condição:

```
if(op==R.id.rb40)
```

Verifico se a opção de 40% foi selecionada, se foi selecionada, realiza o cálculo do salário com o reajuste de 40%. A mesma explicação e válida para o cálculo dos outros reajustes. Agora vamos executar a nossa aplicação. O resultado você vê na figura seguinte:



Aplicação de cálculo de salário em execução

4.4) Desenvolvendo uma aplicação de lista de contatos

Agora vamos fazer uma nova aplicação em Android que consiste em uma aplicação de lista de contatos. Para essa aplicação iremos utilizar um componente chamado **ListView** , que seja bastante útil para esse tipo de situação (quando queremos exibir itens). Toda vez que clicarmos (ou melhor "tocarmos") em um contato na lista, será exibida uma mensagem com o nome do contato selecionado.

Vamos criar agora um novo projeto no Eclipse para Android, conforme os dados abaixo:

Application Name: ListaDeContatos

Project Name: ListaDeContatos

Package Name : com.example.listadecontatos

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

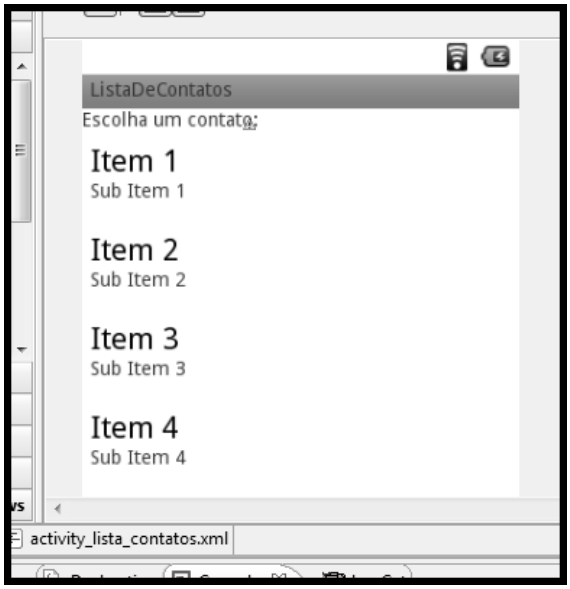
Activity Name: ListaContatosActivity

Layout Name : activity_lista_contatos

Title : Lista de Contatos

Vamos trocar a estrutura de layout padrão para o **LinearLayout**, e no componente **TextView**, dentro da sua propriedade “Text” digite a seguinte frase : “Escolha um contato”.

Em seguida vamos adicionar o componente **ListView (**que se encontra na seção “Composite”). Seguindo o que foi se pedido, a tela da aplicação ficará de acordo com a seguinte figura:



Layout da tela da aplicação em desenvolvimento

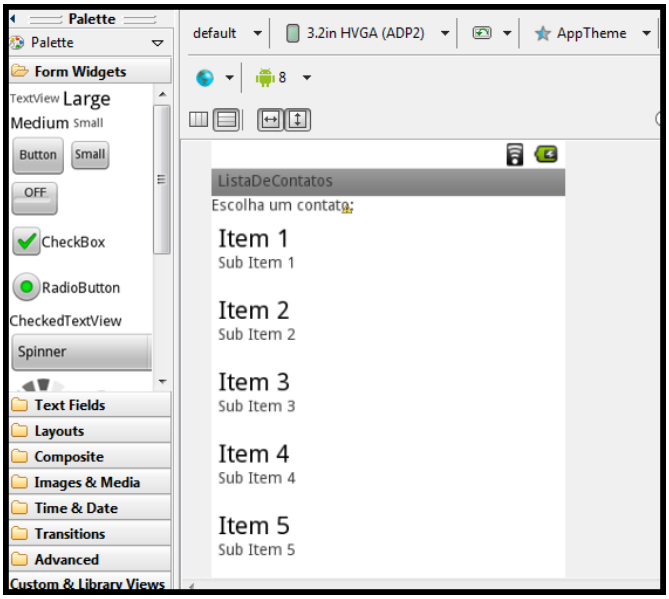
Agora vamos criar um objeto (String Array) que vai armazenar os contatos que serão exibidos no componente, que iremos chamado de “contatos” (criar no arquivo “strings.xml”). Os contatos que estarão nessa lista são : "Aline","Lucas","Rafael","Gabriela" e "Silvana".

Depois de criar os contatos, selecione o objeto **ListView** que você adicionou e altere as seguintes propriedades.

ListView

Propriedade	Valor
Id	lista_contatos
Entries	@array/contatos

Depois de feitas as alterações, a “cara” da aplicação deve estar de acordo com a figura seguinte:



Layout da tela da aplicação

Agora vamos no arquivo “ListaContatosActivity.java” para colocar o seguinte código abaixo (lembre-se antes de salvar o arquivo “activity_lista_contatos.xml”):

```
package com.example.listadecontatos;


import android.os.Bundle;
import android.widget.*;
import android.view.*;
import android.app.*;


public class ListaContatosActivity extends Activity {

    ListView lista_contatos;
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_lista_contatos);

      lista_contatos = (ListView) findViewById
      (R.id.lista_contatos);

      lista_contatos.setOnItemClickListener(new
      AdapterView.OnItemClickListener() {
```

```
      @Override
      public void onItemClick(AdapterView<?> parent,
      View view, int position,long id) {

       String nome = ((TextView) view).getText()
      .toString();



       AlertDialog.Builder dialogo = new
       AlertDialog.Builder(ListaContatosActivity.this)
       dialogo.setTitle("Contato");
       dialogo.setMessage("Contato selecionado: " +
       nome);
   dialogo.setNeutralButton("OK", null);
   dialogo.show();
      }
   });
  }


}
```

Como havia falado (e também como vocês podem conferir no código acima), quando se clica em um item, o sistema mostra uma mensagem do item selecionado (no caso, o nome contato selecionado). Isso é conseguido fazendo uso da interface **OnItemClickListener**, como mostra a instrução abaixo:

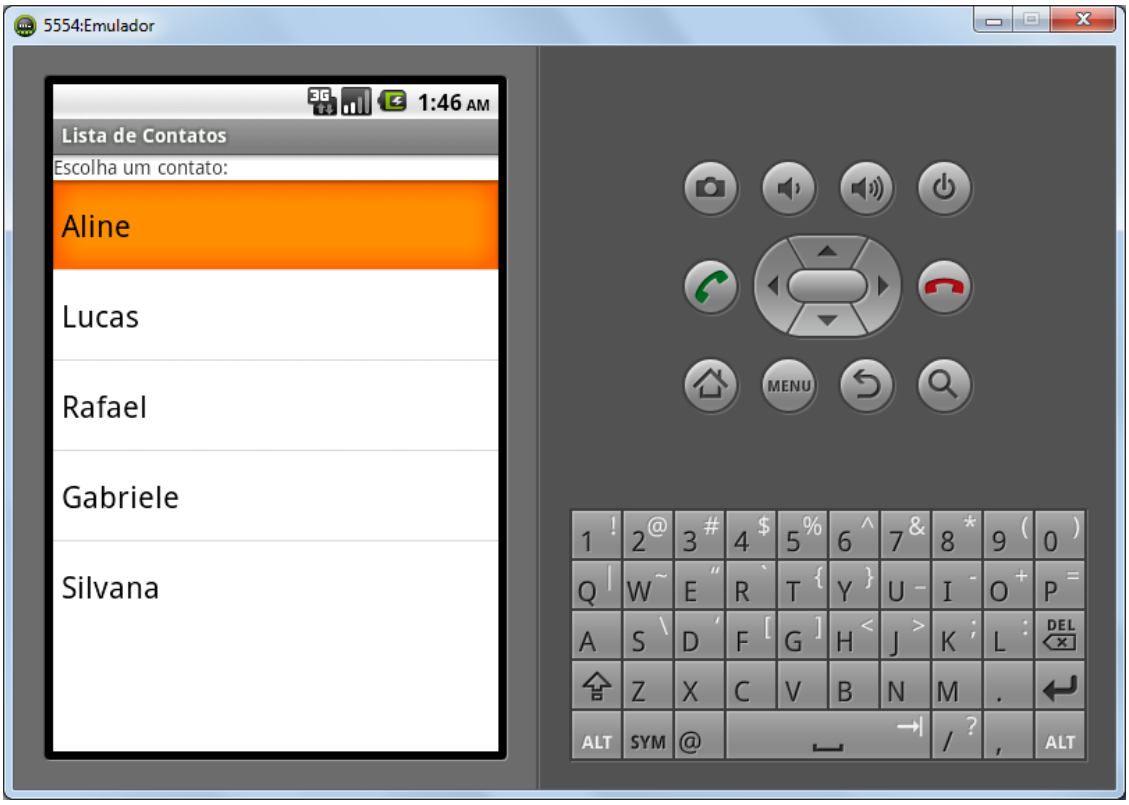
```
lista_contatos.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent,
View view, int position,long id) {

   String nome = ((TextView) view).getText()
   .toString();



   AlertDialog.Builder dialogo = new
   AlertDialog.Builder(ListaContatosActivity.this)
   dialogo.setTitle("Contato");
   dialogo.setMessage("Contato selecionado: " + nome);
   dialogo.setNeutralButton("OK", null);
   dialogo.show();
      }
   });
```

Vamos analisar alguns trechos do código. A linha de comando:

```
     String nome = ((TextView) view).getText().toString();
```

Guarda na variável *"nome"* o conteúdo retornado pelo objeto *"view"* (que contém o contato selecionado). Como o conteúdo precisa ser retornado para a variável que é do tipo **String**, foi preciso converte-lo em **TextView** para que o conteúdo fosse retornado em uma **String** (através do método **toString** situado em **getText**).

Vamos executar a aplicação. O resultado você vê na figura abaixo:



Aplicação de lista de contatos em execução

5.6) Desenvolvendo uma aplicação que visualiza imagens

Agora vamos desenvolver uma aplicação básica que visualiza imagens através do uso o componente **ImageView**. Vamos criar um projeto com os seguintes dados abaixo:

Application Name: VisualizadorDeImagens

Project Name: VisualizadorDeImagens

Package Name : com.example.visualizadordeimagens

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: VisualizadorImagensActivity

Layout Name : activity_visualizador_imagens

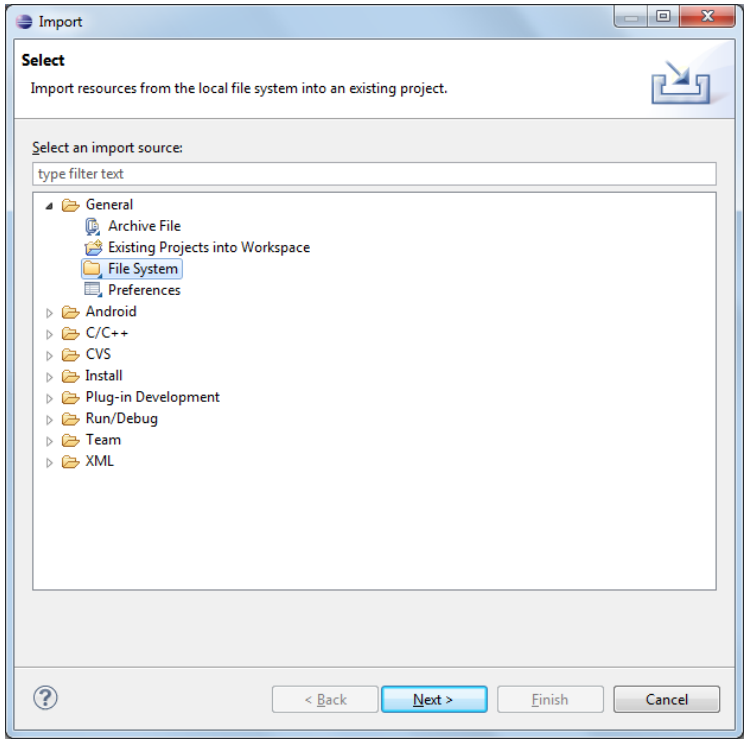
Title : Visualizador de Imagens

Depois de criado o projeto vamos trocar a estrutura de layout para o **LinearLayout** em seguida "apague" o componente **TextView** da tela.

Antes de iniciarmos a codificação do programa, quero que você coloque duas imagens JPEG (com a extensão .jpg), dentro da pasta "res/drawable-mdpi" (para esse projeto usei duas imagens chamadas "foto1.jpg" e "foto2.jpg").

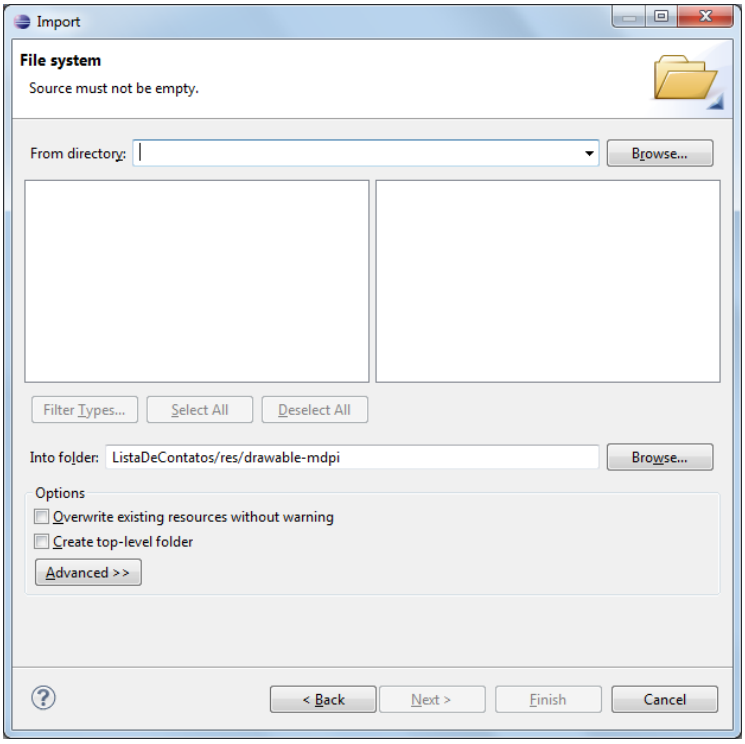
Irei mostrar aqui duas formas de se realizar essa tarefa. Começarei pela primeira forma, que considero a mais tradicional de todas.

Para importar um arquivo, clique com o botão direito do mouse sobre a pasta "res/drawable-mdpi" e selecione "Import", depois selecione "File System" (Que se encontra dentro da pasta "General", conforme mostra a figura abaixo) e em seguida clique em "Next".



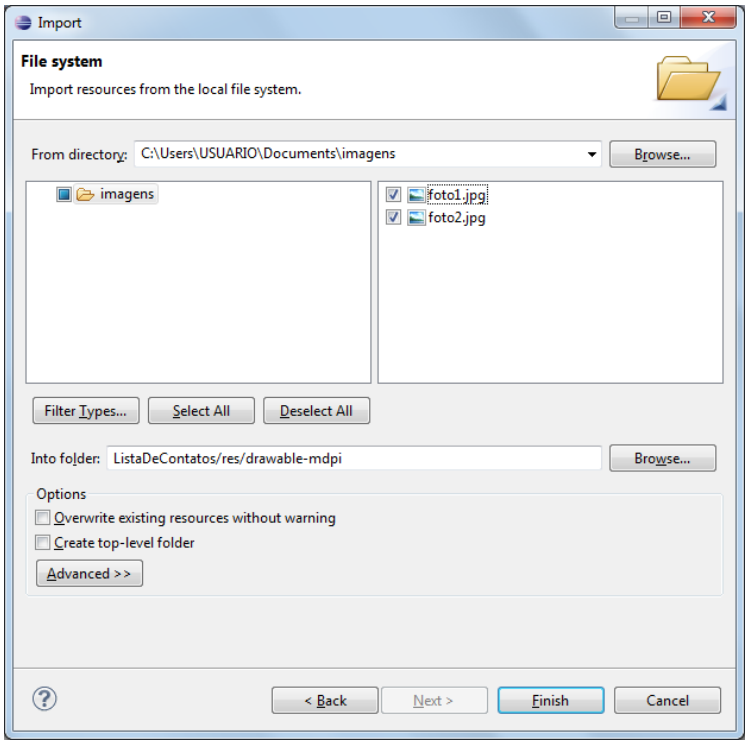
Selecionando a opção "File System"

Após clicar em “Next” será exibida a caixa de diálogo como demonstra a figura abaixo:



Caixa de diálogo “File System”

Clique no botão “Browse...” para selecionar o diretório onde se encontram as imagens. Feito isso, marque os dois arquivos (imagens) para que eles sejam importados para a pasta “res/drawable-mdpi” . Veja a figura abaixo:

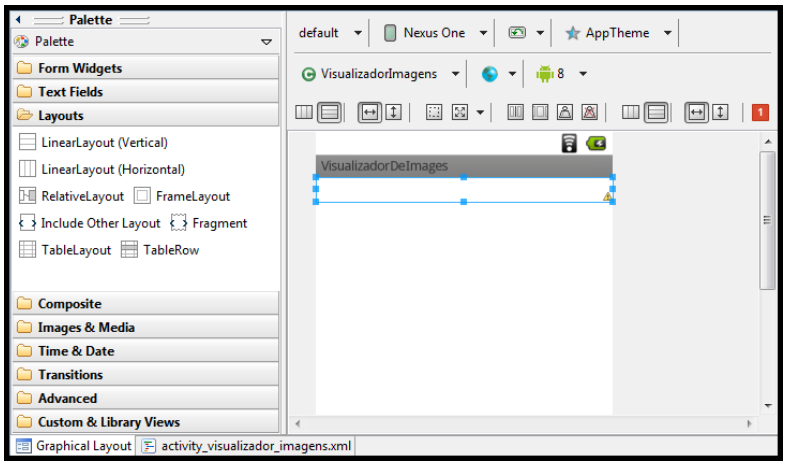


Importando as imagens para o projeto

Depois disso, é só clicar em “Finish” para importar as imagens para o projeto.

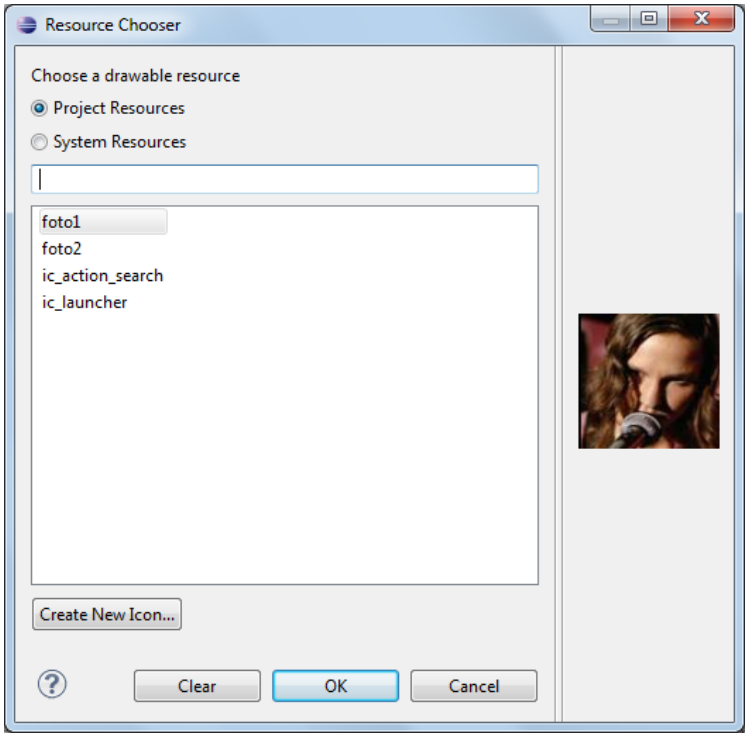
A segunda forma, que diria que é a mais fácil de todas, é você ir até o diretório onde se encontram as imagens , para que você em seguida possa selecioná-las, e logo após copiá-las (o famoso “Ctrl+C”). Feito isso vá até o projeto que criamos para selecionarmos o diretório “drawable-mpdi” para colarmos as imagens dentro da pasta (simples não ?).

Vamos adicionar dentro da tela da nossa aplicação uma estrutura **LinearLayout** (Horizontal), que se encontra na guia “Layouts”, simplesmente arrastando o componente para a tela da aplicação. O resultado você confere na figura abaixo:



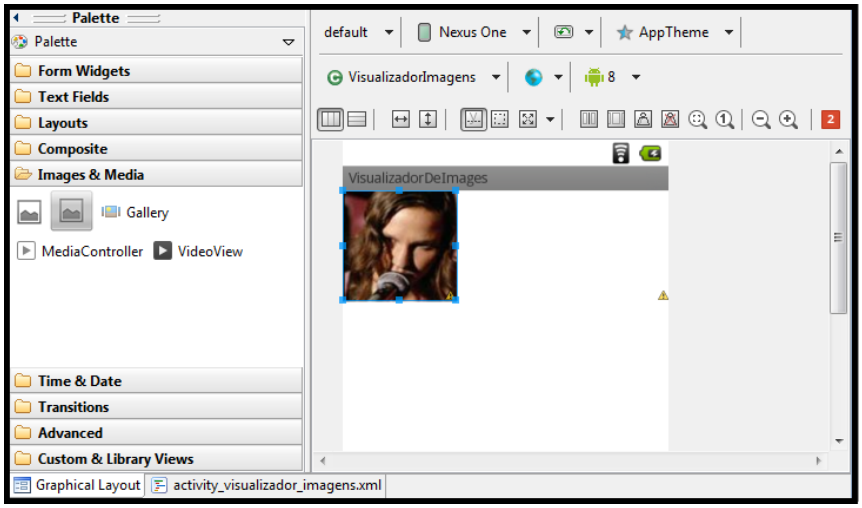
Estrutura LinearLayout inserida

Agora dentro da estrutura **LinearLayout** que adicionamos acima, vamos inserir o componente **ImageView** (que se encontra na guia “Images & Media”). Quando colocamos o componente no local desejado da tela, é exibido a seguinte caixa de diálogo:



Caixa de diálogo – Resource Chooser

Nesta caixa de diálogo escolhemos a imagem que o nosso componente vai assumir inicialmente. Iremos escolher a imagem chamada “foto1” (conforme é demonstrado acima). Depois de escolher a imagem clique em “OK”. Veja o resultado em seguida:



Resultado da operação

Agora vamos alterar a propriedade do componente **ImageView** conforme abaixo:

ImageView

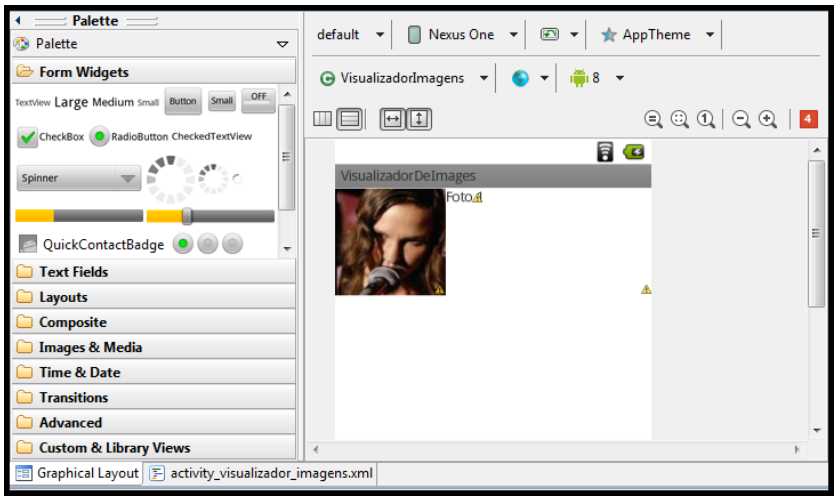
Propriedade	Valor
Id	imgfoto

Agora vamos adicionar um componente **TextView** que ficará ao lado da imagem. Altere suas propriedades conforme a tabela abaixo:

TextView

Propriedade	Valor
Id	txtinformacao
Text	Foto 1

Seguindo os passos acima, o resultado do layout deve ficar de acordo com a figura abaixo:



Layout da aplicação

Agora vamos adicionar na sequência dois componentes do tipo **Button**, só que esses dois componentes vão estar dentro da tela da aplicação e fora (e também abaixo) da estrutura de layout que adicionamos. Segue abaixo as propriedades que precisam ser modificadas:

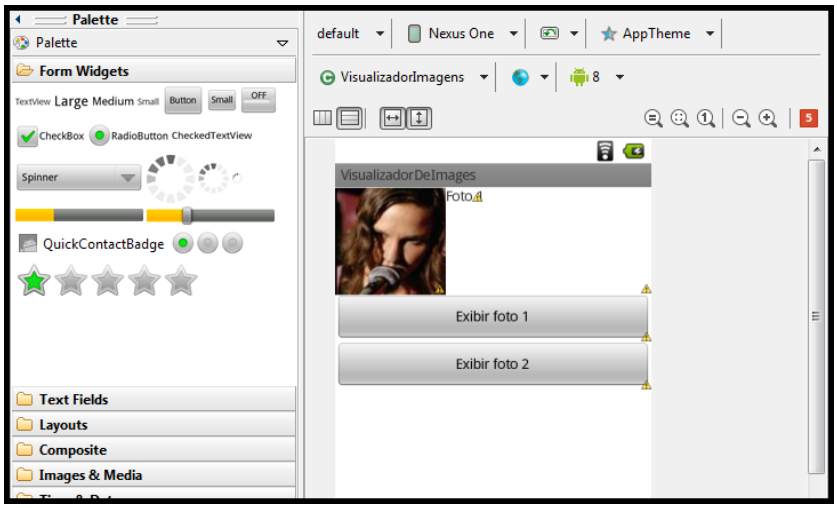
Button

Propriedade	Valor
Id	btfoto1
Text	Exibir foto 1
Width	fill_parent

Button

Propriedade	Valor
Id	Btfoto2
Text	Exibir foto 2
Width	fill_parent

Depois de seguir todos os passos descritos acima, a aplicação tem que estar de acordo com a figura abaixo:



Layout da aplicação

Agora vamos no arquivo “VisualizadorImagensActivity.java” para colocarmos o código em seguida (lembre-se de salvar o arquivo “activity_visualizador_imagens.xml” antes de escrever o código):

```
package com.example.visualizadordeimages;

import android.os.Bundle;
import android.app.Activity;
import android.widget.*;
import android.view.*;


public class VisualizadorImagensActivity extends Activity {

  ImageView imgfoto;

  Button btfoto1,btfoto2;

  TextView txtinformacao;

  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);

     setContentView
     (R.layout.activity_visualizador_imagens);

     imgfoto = (ImageView) findViewById(R.id.imgfoto);

     btfoto1 = (Button) findViewById(R.id.btfoto1);
     btfoto2 = (Button) findViewById(R.id.btfoto2);

     txtinformacao = (TextView) findViewById
     (R.id.txtinformacao);

     btfoto1.setOnClickListener(new View.
     OnClickListener() {
```

```
      @Override
      public void onClick(View arg0) {

        imgfoto.setImageResource
        (R.drawable.foto1);

        txtinformacao.setText("Foto 1");

       }
      });

   btfoto2.setOnClickListener(new View.OnClickListener()
    {

      @Override
      public void onClick(View arg0) {

         imgfoto.setImageResource
         (R.drawable.foto2);
         txtinformacao.setText("Foto 2");

        }
      });

    }


}
```

Agora vamos analisar alguns trechos de códigos. Vamos no evento Click referente a abertura da primeira imagem. O código:

```
imgfoto.setImageResource(R.drawable.foto1);
```

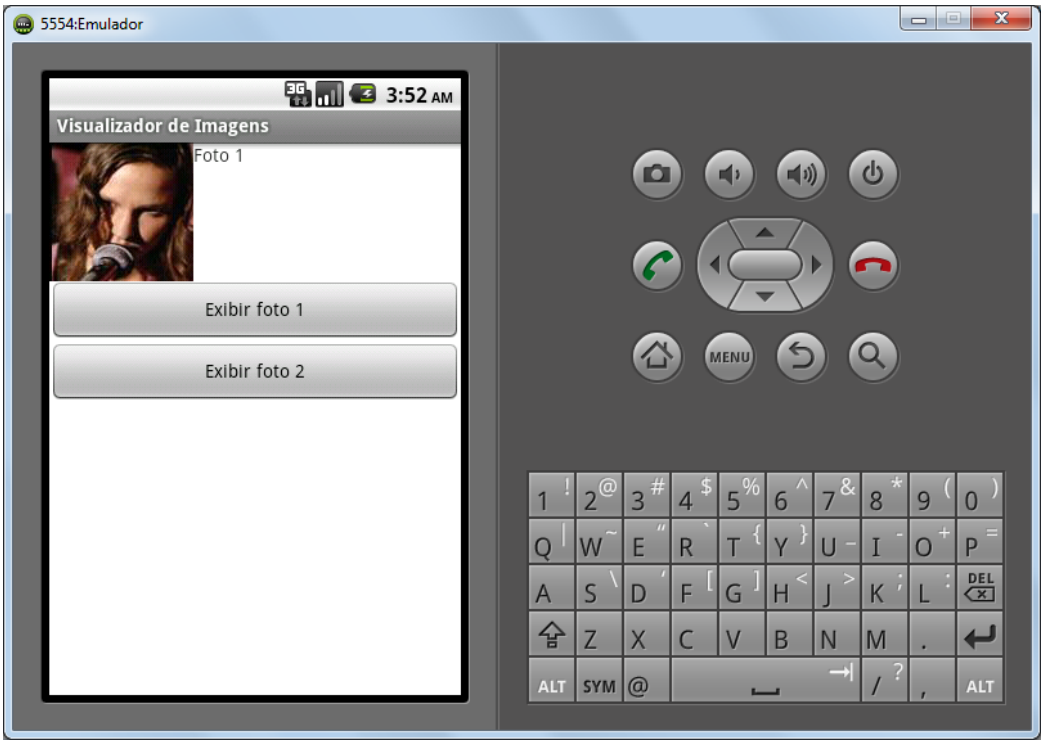
É responsável por abrir a imagem "foto1.jpg" e exibi-la no componente. Observe que foi passado o parâmetro *"R.drawable.foto1"* onde *"drawable"* corresponde a pasta e "*foto1"* corresponde ao arquivo *"foto1.jpg"*. Logo após vem o código:

```
txtinformacao.setText("Foto 1");
```

Cuja finalidade é mudar o título da **TextView** , de acordo com a String passada como parâmetro.

O comentário acima é o mesmo para o segundo botão referente à abertura da segunda imagem.

Vamos executar a nossa aplicação. O resultado você vê nas imagens abaixo:



Aplicação de visualização de imagens em execução

Capitulo 4 Trabalhando com mais de uma tela em uma aplicação

Até agora as aplicações que desenvolvemos tinham somente uma única tela, mas, sabemos que algumas aplicações possuem normalmente mais de uma tela. A partir de agora iremos aprender como inserir e gerenciar várias telas em uma aplicação Android através dos exemplos que serão demonstrados nesse capítulo.

Para começarmos, vamos criar um novo projeto Android com os seguintes dados abaixo:

Application Name: TrocaDeTelas

Project Name: TrocaDeTelas

Package Name : com.example.trocadetelas

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: TrocaTelasActivity

Layout Name : activity_tela_principal

Title : Troca de Telas

Altere a estrutura de layout da sua aplicação para o **LinearLayout** e em seguida altere o componente **TextView** de acordo com a tabela abaixo.

TextView

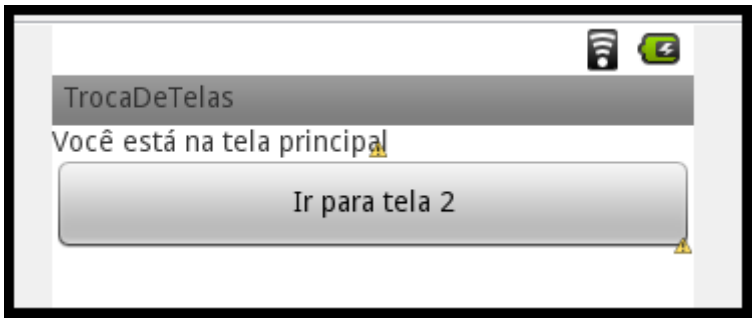
Propriedade	Valor
Padding	(Deixar em branco)
Text	Você está na tela principal

Agora adicione um componente **Button** e modifique as seguintes propriedades:

Button

Propriedade	Valor
Id	bttela2
Layout width	fill_parent
Text	Ir pata tela 2

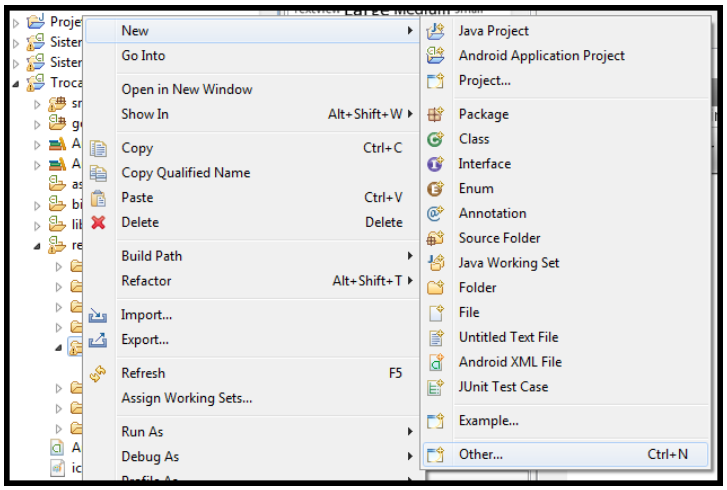
Seguindo os passos acima, a aplicação deve estar de acordo com a figura abaixo:



Layout da tela 1

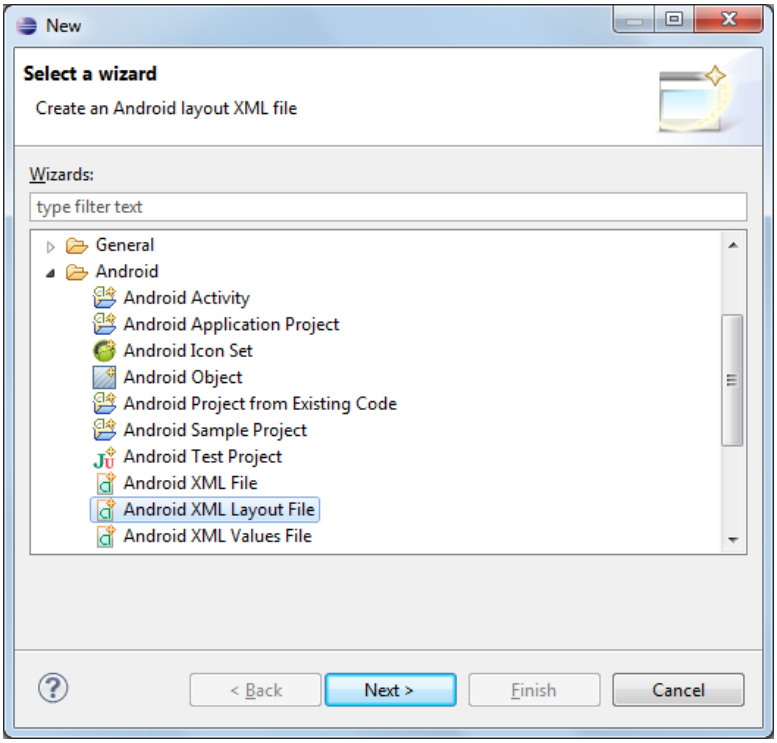
Nossa primeira tela está pronta, muito bem! Agora vamos criar uma nova tela para a nossa aplicação. O nome do arquivo que vai representar a segunda tela da nossa aplicação vai se chamar "tela2.xml" (um arquivo XML). Conforme já foi explicado (e explico novamente aqui), todos os arquivos que representam a tela da aplicação devem estar dentro do diretório "layout" (situado dentro da pasta "res" do projeto), logo, vamos criar o nosso arquivo dentro desse diretório.

Para criarmos um novo arquivo XML dentro do diretório "layout" basta clicar com o botão direito sobre a pasta e em seguida clicar em "New" e logo após "Other", confira na figura seguinte:



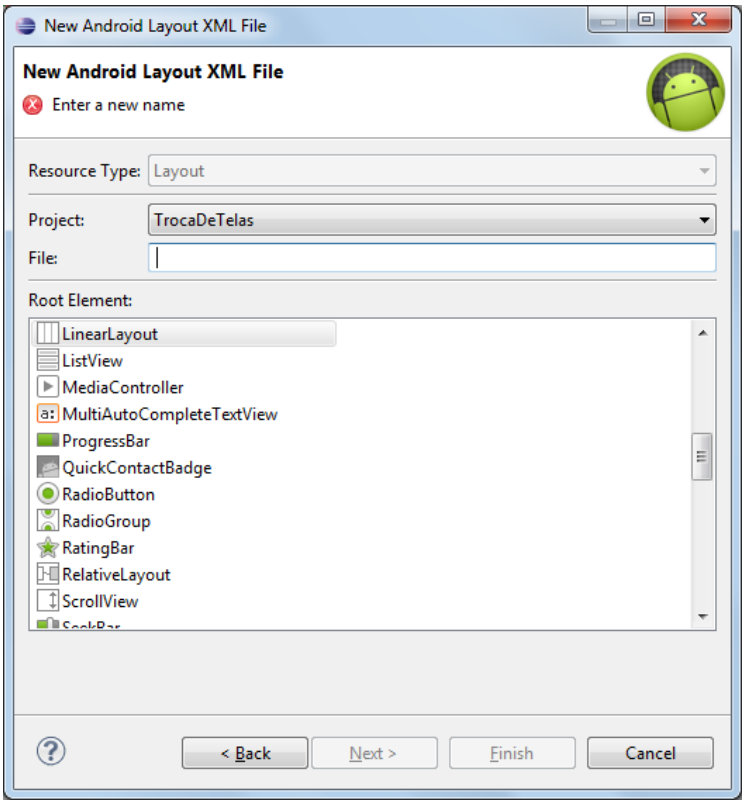
Criando um arquivo XML

Feito isso será aberto uma tela com várias pastas, você irá expandir a pasta “Android” e em seguida vai selecionar a opção “Android Layout XML File”, conforme mostra a figura seguinte:



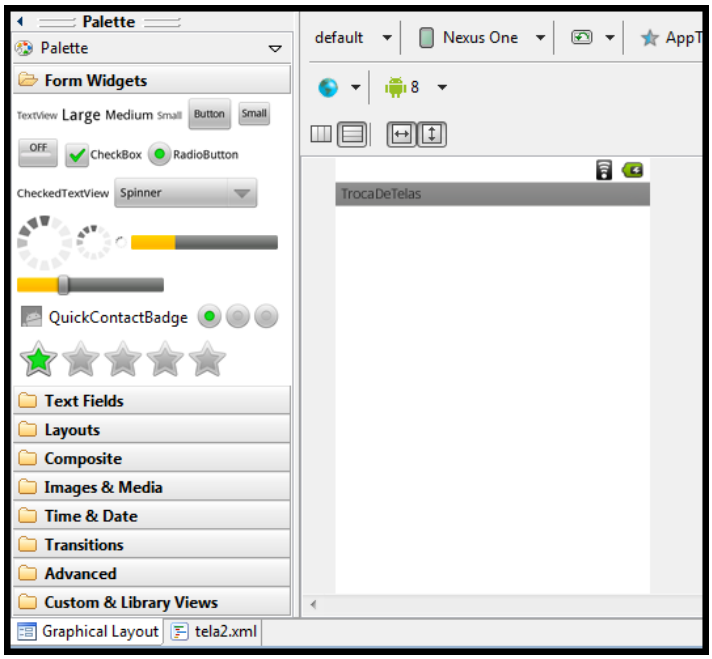
Android XML Layout File

Após selecionar a opção acima clique em “Next” que em seguida será aberta a tela seguinte:



New Android XML Layout File

Por padrão a estrutura de layout padrão selecionada é **LinearLayout** (que será a estrutura que iremos usar). Agora no campo “File” digite “tela2” e em seguida clique em “Finish” para que o arquivo seja gerado. Veja o resultado na figura seguinte:



Tela de layout em branco

Por padrão quando criamos a tela da aplicação utilizando o **LinearLayout**, ele cria com a orientação "vertical" (essa é que vamos utilizar, portanto, não vamos mexer na estrutura).

Agora vamos adicionar os seguintes componentes, na sequência:

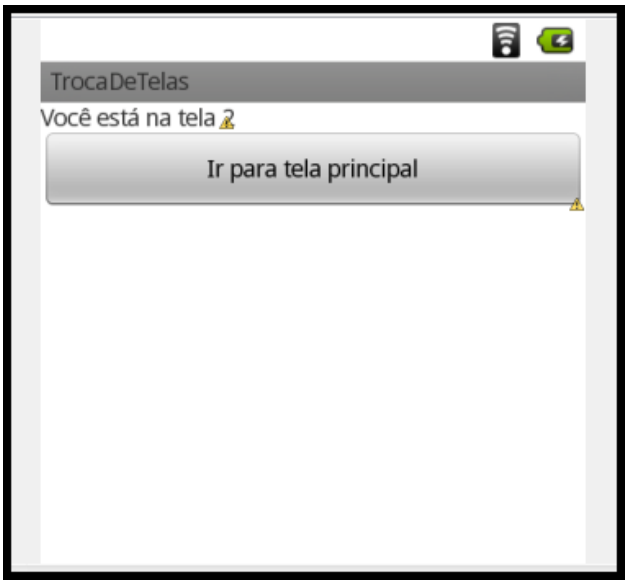
TextView

Propriedade	Valor
Text	Você está na tela 2

Button

Propriedade	Valor
Id	bttelaprincipal
Width	fill_parent
Text	Ir pata tela principal

Seguindo os passos acima, o layout do arquivo "tela2.xml" deve estar de acordo com a figura abaixo:



Layout da tela 2

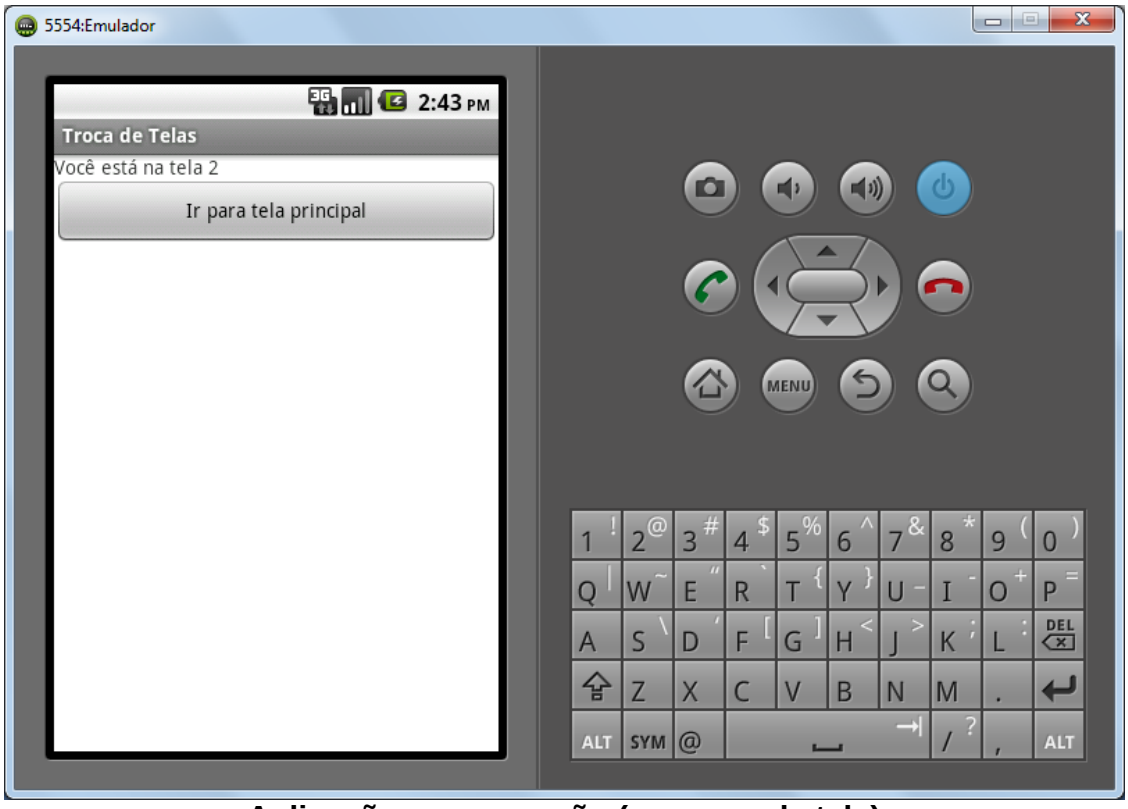
Agora vamos no arquivo “TrocaTelasActivity.java” para digitarmos o código abaixo:

```
package com.example.trocadetelas;

import android.os.Bundle;
import android.app.Activity;
import android.widget.*;
import android.view.*;

public class TrocaTelasActivity extends Activity {

      Button bttelaprincipal, bttela2;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        CarregarTelaPrincipal();
    }
    public void CarregarTelaPrincipal()
    {
      setContentView(R.layout.tela_principal);
      bttela2 = (Button) findViewById(R.id.bttela2);
      bttela2.setOnClickListener(new
       View.OnClickListener() {

         @Override
         public void onClick(View v) {

               CarregarTela2();

           }
         });
    }

    public void CarregarTela2()
    {
```

```
        setContentView(R.layout.tela2);
        bttelaprincipal = (Button) findViewById
        (R.id.bttelaprincipal);
        bttelaprincipal.setOnClickListener(new
         View.OnClickListener() {

              @Override
               public void onClick(View v) {

                  CarregarTelaPrincipal();

               }
            });
    }
}
```

Observem que nesta classe eu criei dois métodos : **CarregaTelaPrincipal** e **CarregaTela2**. Para toda aplicação que utilize mais de um layout (tela), o carregamento dos layouts e de seus respectivos widgets devem estar separados em funções desenvolvidas para esse propósito. Logo, o método **CarregaTelaPrincipal** carrega o layout principal e seus respectivos componentes, o mesmo válido para o método **CarregaTela2**, que carrega o layout da tela 2 e seus respectivos componentes. Feito isso, execute a aplicação. Veja o resultado abaixo:

Aplicação em execução (na tela principal)



Aplicação em execução (na segunda tela)

Capitulo 5 Propriedades e eventos dos componentes trabalhados

Neste capitulo eu irei mostrar e descrever as propriedades e eventos de todos os componentes que trabalhamos neste material.

Widget TextView

- Propriedades

Propriedade	Em XML	Em Java
`Text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`Text Color`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor de texto.		

Propriedade	Em XML	Em Java
`Background`	`android:background`	`setBackGroundColor(Color c)`
Nessa propriedade , você define o cor de fundo do componente exibido. Valor: #000000 até #FFFFFF.		

Propriedade	Em XML	Em Java

`Text Size`	`android:textSize`	`setTextSize(float tamanho) ou`
		
		`setTextSize(int unidade, int tamanho)`

Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (polegadas) e etc.

Propriedade	Em XML	Em Java
`Typeface`	`android:typeface`	`setTypeface(Typeface fonte)`
Essa propriedade serve para definir uma fonte ao texto (normal,sans,serif,monospace).		

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Widget EditText

- Propriedades

Propriedade	Em XML	Em Java
`Text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`Text Color`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
`Background`	`android:background`	`setBackGroundColor(Color c)`
Nessa propriedade , você define o cor de fundo do componente exibido. Valor: #000000 até #FFFFFF.		

Propriedade	Em XML	Em Java
`Capitalize`	`android:capitalize`	
Essa propriedade serve para definir o tipo capitalização das palavras. Por padrão, o valor e “none”(nenhum). Os possíveis valores para essa propriedade são : “words”,”sentences” e ”characters”		

Propriedade	Em XML	Em Java
Password	android:password	
Com essa propriedade você habilita a digitação de senhas. O valor padrão desse atributo é “false”.		

Propriedade	Em XML	Em Java
Text Size	android:textSize	setTextSize(float tamanho) ou setTextSize(int unidade, int tamanho)
Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (polegadas) e etc.		

Propriedade	Em XML	Em Java
Typeface	android:typeface	setTypeface(Typeface fonte)
Essa propriedade serve para definir uma fonte ao texto. Os possíveis valores são : “normal”,”monospace”,”sans” e “serif”.		

Propriedade	Em XML	Em Java
Hint	android:hint	setHint(CharSequence c)
define uma mensagem que aparecerá quando a EditText estiver vazia.		

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento
setOnClickListener	OnClickListener	onClick(View v)
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
setOnKeyListener	OnKeyListener	onKey(View v,int KeyCode, KeyEvent event)
Esse evento é disparado toda vez que a tecla é acionada, disparando o método onKey.		

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnFocusChangeListener`	`OnFocusChangeListener`	`onFocusChange(View v, boolean hasFocus)`
Esse método é disparado toda vez quando um componente EditText ganha ou perde foco.		

Widget Button

- Propriedades

Propriedade	Em XML	Em Java
`Text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`Text Color`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
`Text size`	`android:textSize`	`setTextSize(float tamanho)` ou `setTextSize(int unidade, int tamanho)`
Define o tamanho do texto. O tamanho da fonte pode ser especificado em várias notações : px (pixels),sp(scaled-pixels) , mm(milímetros), in (polegadas) e etc.		

Propriedade	Em XML	Em Java
`Typeface`	`android:typeface`	`setTypeface(Typeface fonte)`
Essa propriedade serve para definir uma fonte ao texto. Os possíveis valores são : “normal”,”monospace”,”sans” e “serif”.		

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnKeyListener`	`OnKeyListener`	`onKey(View v,int KeyCode, KeyEvent event)`
Esse evento é disparado toda vez que a tecla é acionada, disparando o método onKey.		

Widget CheckBox/RadioButton

- Propriedades

Propriedade	Em XML	Em Java
`Text`	`android:text`	`setText(CharSequence c)`
Nessa propriedade, você define o texto a ser exibido na tela.		

Propriedade	Em XML	Em Java
`Text color`	`android:textColor`	`setTextColor(Color c)`
Nessa propriedade, você define a cor do texto.		

Propriedade	Em XML	Em Java
`Checked`	`android:checked`	`setChecked(boolean estado)`
Nessa propriedade você define o estado do CheckBox, se estará marcado (true) ou não (false).		

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento

`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnCheckedChangeListener`	`OnCheckedChange Listener`	`onCheckedChanged (CompoundButton cb,boolean b)`
Esse evento será disparado toda vez que o estado do CheckBox for modificado, ou seja, marcado ou desmarcado, disparando o método onCheckedChanged.		

Widget ListView

- Propriedades

Método	Descrição
`setAdapter(SpinnerAdapter a)`	Nesse método você define os elementos que irão compor esse componente através de um vetor (array).
`int getSelectedPosition()`	Essa função retorna a posição do elemento selecionado. Por exemplo, se for o primeiro elemento, retorna 0, se for o segundo, retorna 1 e assim sucessivamente.
`Object getSelectedItem()`	Essa função retorna em um tipo Object, o item selecionado.
`Object getItemAtPosition(int posicao)`	Retorna em um tipo Object o elemento de uma determinada posição, passada como parâmetro.

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnItemClickListener`	`OnItemClickListener`	`onItemClick (AdapterView<?> a, View v, int I, long l)`
Esse evento será disparado toda vez que um derminado item for clicado, disparando o método onItemClick.		

Método que define o evento	Evento	Métodos relacionados ao evento

`setOnItemSelectedListener`	`OnItemSelectedListener`	`onItemSelected(AdapterView av, View v,int posição,long id)`
		
		`onNothingSelected(AdapterView av)`

Esse evento será disparado toda vez que um derminado item for selecionado, disparando o método onItemSelected. Caso nenhum item seja selecionado, será disparado o método onNothingSelected.		

Widget ImageView

- Propriedades

Propriedade	Em XML	Em Java
`Src`	`android:src`	`setImageResource(int Id)`
Nessa propriedade, você define a imagem que será exibida na tela.		

Método	Descrição
`setImageURI(Uri link)`	Esse método é similar ao método acima, sendo que aqui você especifica o Uri (como se fosse um link de internet) como caminho de localização da imagem.

- Eventos

Método que define o evento	Evento	Métodos relacionados ao evento
`setOnClickListener`	`OnClickListener`	`onClick(View v)`
Esse evento é disparado toda vez que o componente for clicado, disparando o método onClick.		

Conclusão a respeito do material

Nesta apostila vimos de forma bem básica e introdutória como desenvolver aplicações para Android para algumas situações. Começamos vendo um pouco sobre a plataforma Android, como ela surgiu e tudo mais. Aprendemos a instalar o Eclipse e os plugins necessários para o funcionamento do Android, incluindo o SDK, e em seguida aprendemos a construir algumas pequenas aplicações para Android, como uma calculadora básica, um aplicativo de compras, um aplicativo de cálculo de salário e etc.

Se você quiser uma abordagem "mais completa" de como desenvolver aplicações para Android, adquira a "Apostila de Android – Programando Passo a Passo" Completa, efetuando o pagamento do seu valor através do **PagSeguro.** Visite o site **www.apostilaandroid.net** para mais informações à respeito da Apostila de Android "completa".

Espero que esse material lhe tenha sido útil.

Abraços